Personal Computer PC*jr*
Hardware Reference
Library

Technical Reference

6322963

Update to the IBM PC*jr* Technical Reference

This update contains information that is pertinent to the IBM PC Compact Printer.

Insert the pages contained in this package into your IBM PC*jr* Technical Reference.

The following pages replace existing pages in your Technical Reference.

- Table of Contents (vii, viii, ix, and x)
- 3-3 and 3-4
- A1 and A2
- B1 and B2
- D-7 and D-8
- Index-1 through Index-24

Add the following pages to your Technical Reference.

- Tab Index xi, xii, xiii, xiv
- 3-133 through 3-150
- B-47

IBM

Personal Computer PC*jr*
Hardware Reference
Library

Technical Reference

First Edition Revised (November 1983)

Changes are periodically made to the information herein; these changes will be incorporated in new editions of this publication.

Products are not stocked at the address below. Requests for copies of this product and for technical information about the system should be made to your authorized IBM Personal Computer dealer.

A Reader's Comment Form is provided at the back of this publication. If this form has been removed, address comments to IBM Corporation, Personal Computer, P.O. Box 1328-C, Boca Raton, Florida 33432. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligations whatever.

FEDERAL COMMUNICATIONS COMMISSION RADIO FREQUENCY INTERFERENCE STATEMENT

Warning: This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

INSTRUCTIONS TO USER

This equipment generates and uses radio frequency energy and if not installed and used properly, i.e., in strict accordance with the operating instructions, reference manuals, and the service manual, may cause interference to radio or television reception. It has been tested and found to comply with the limits for a Class B computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a residential installation.

If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna.
- Relocate the equipment with respect to the receiver.
- Move the equipment away from the receiver.
- Plug the equipment into a different outlet so that equipment and receiver are on different branch circuits.
- Ensure that side option mounting screws, attachment connector screws, and ground wires are tightly secured.
- If peripherals not offered by IBM are used with this equipment, it is suggested that you use shielded, grounded cables with in-line filters, if necessary.

If necessary, consult your dealer service representative for additional suggestions.

The manufacturer is not responsible for any radio or TV interference caused by unauthorized modifications to this equipment. It is the responsibility of the user to correct such interference.

> **CAUTION**
> This product is equipped with a UL listed and CSA-certified plug for the user's safety. It is to be used in conjunction with a properly grounded 115 Vac receptacle to avoid electrical shock.

Preface

The IBM PC*jr* Technical Reference manual describes the hardware design and provides interface information for the IBM PC*jr*. This publication also has information about the basic input/output system (BIOS) and programming support.

The information in this publication is both descriptive and reference oriented, and is intended for hardware and software designers, programmers, engineers, and interested persons who need to understand the design and operation of the IBM PC*jr* computer.

You should be familiar with the use of the IBM PC*jr*, and understand the concepts of computer architecture and programming.

This manual has five sections:

Section 1: "Introduction" is an overview of the basic system and available options.

Section 2: "Base System" describes each functional part of the base system. This section also has specifications for power, timing, and interfaces. Programming considerations are supported by coding tables, command codes, and registers.

Section 3: "System Options" describes each available option using the same format as Section 2: "Base System."

Section 4: "Compatibility with the IBM Personal Computer Family" describes programming concerns for maintaining compatibility between the IBM PC*jr* and the other IBM Personal Computers.

Section 5: "System BIOS and Usage" describes the basic input/output system (BIOS) and its use. This section also contains the software interrupt listing, a system memory map, descriptions of vectors with special meanings, and a set of low-storage maps. In addition, keyboard encoding and usage is discussed.

This publication has four appendixes:

Appendix A: "ROM BIOS Listing"
Appendix B: "Logic Diagrams"
Appendix C: "Characters, Keystrokes, and Color"
Appendix D: "Unit Specifications"

Prerequisite Publication:

Guide to Operations part number 1502291

Guide to Operations part number 1502292

Suggested Reading:

IBM PCjr Hands on BASIC part number 1504702
IBM PCjr BASIC Reference Manual part number 6182371
Disk Operating System (DOS) part number 6024061
Hardware Maintenance and Service Manual part number 1502294
Macro Assembler part number 6024002

Related publications are listed in "Bibliography."

Contents

Notes:

TAB INDEX

Notes:

Notes:

SECTION 1. INTRODUCTION

Contents

Notes:

Introduction

The system unit, a desk top transformer, and a cordless keyboard make up the hardware for the PC*jr* base system.

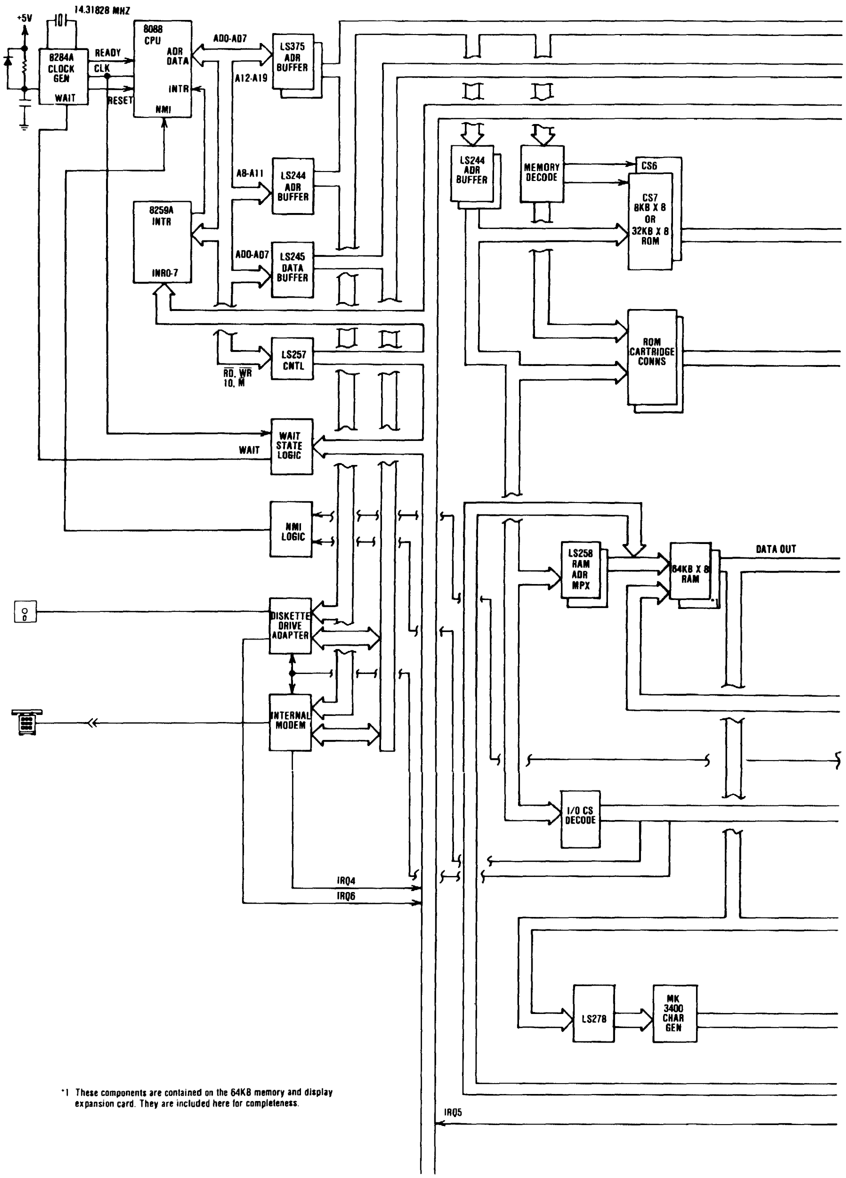
The following options are available for the base system:

- IBM PC*jr* 64KB Memory and Display Expansion
  - The 64KB Memory and Display Expansion enables the user to work with the higher density video modes while increasing the system's memory size by 64K Bytes to a total of 128K Bytes.
- IBM PC*jr* Diskette Drive Adapter
  - The IBM PC*jr* Diskette Drive Adapter permits the attachment of the IBM PC*jr* Diskette Drive to the IBM PC*jr* and resides in a dedicated connector on the IBM PC*jr* system board.
- IBM PC*jr* Diskette Drive
  - The IBM PC*jr* Diskette Drive is double-sided with 40 tracks for each side, is fully self-contained, and consists of a spindle drive system, a read positioning system, and a read/write/erase system.
- IBM PC*jr* Internal Modem
  - The IBM PC*jr* Internal Modem is an adapter that plugs into the PC*jr* system board modem connector and allows communications over standard telephone lines.

- IBM PC*jr* Parallel Printer Attachment
  - The IBM PC*jr* Parallel Printer Attachment is provided to attach various I/O devices that accept eight bits of parallel data at standard TTL logic levels. It attaches as a feature to the right side of the system unit.
- IBM Personal Computer Graphics Printer
  - IBM Graphics Printer is an 80 cps (characters-per-second), self-powered, stand-alone, tabletop unit.
- IBM PC*jr* Joystick
  - The IBM PC*jr* Joystick is an input device to provide the user with two-dimensional positioning-control. Two pushbutton switches on the joystick give the user additional input capability.
- IBM Color Display
  - The IBM Color Display is a Red/Green/Blue /Intensity (RGBI) Direct-Drive display, that is independently housed and powered.
- IBM Connector for Television
  - The IBM Connector for Television allows a TV to be connected to the IBM PC*jr* system.
- IBM PC*jr* Keyboard Cord
  - The IBM PC*jr* Keyboard Cord option is used to connect the IBM PC*jr* Cordless Keyboard to the system board.

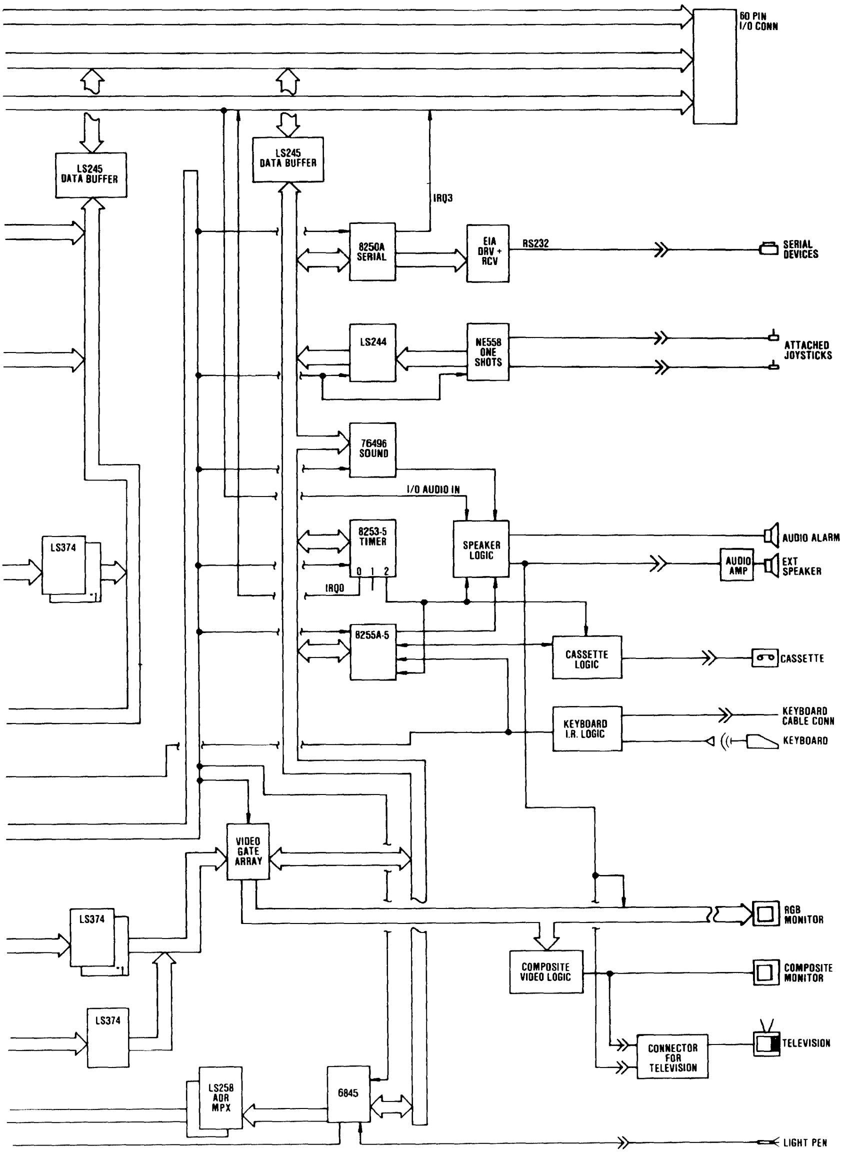
- IBM PC*jr* Adapter Cable for Serial Devices
  - This option is an adapter cable that allows connection of serial devices to the IBM PC*jr* system board.
- IBM PC*jr* Adapter Cable for Cassette
  - This option is an adapter cable that allows a cassette recorder to be connected to the IBM PC*jr*.
- IBM PC*jr* Adapter Cable for Color Display
  - This adapter cable allows the IBM Color Display to be connected to the IBM PC*jr*.

The following is a block diagram of the IBM PC*jr* system.



*1 These components are contained on the 64KB memory and display expansion card. They are included here for completeness.

System Block Diagram (Sheet 1 of 2)



*1 These components are contained on the 64KB memory and display expansion card. They are included here for completeness.

System Block Diagram (Sheet 2 of 2)

Notes:

SECTION 2. BASE SYSTEM

Contents

Notes:

Introduction

The PC*jr* base-system hardware consists of the system unit, a 62-key cordless-keyboard, and a power transformer.

The PC*jr* system board is the center of the PC*jr* system unit. The system board fits horizontally in the base of the system unit and is approximately 255 mm by 350 mm (10 inches by 13.8 inches). It is double-sided, with an internal-power/ground plane. Low voltage ac power enters the power supply adapter, is converted to dc voltage, and enters the system board through the power supply adapter edge-connector. Other system board connectors provide interfaces for a variety of input/output (I/O) devices and are individually keyed to prevent improper installation. The following is a list of these connectors:

- 64KB Memory and Display Expansion Connector
- Diskette Drive Adapter Connector
- Internal Modem Connector
- Infra-Red (IR) Link Receiver Board Connector
- Program Cartridge Connectors (2)
- I/O Channel Expansion Connector
- Serial Port (RS232) Connector (with optional adapter cable)
- Direct Drive (RGBI) Video Connector
- Composite Video Connector
- IBM Connector for Television Connector (external RF modulator)
- Light Pen Connector
- External Audio Connector
- IBM PC*jr* Keyboard Cord Connector
- Cassette Connector (with optional adapter cable)
- IBM PC*jr* Attachable Joystick Connectors (2)

The system board consists of seven functional subsystems: the processor subsystem and its support elements, the read-only (ROM) subsystem, the read/write (R/W) subsystem, the audio subsystem, the video subsystem, the games subsystem, and the I/O channel. All are described in this section.

The nucleus of the system board is the Intel 8088 microprocessor. This processor is an 8-bit external bus version of Intel's 16-bit 8086 processor, and is software-compatible with the 8086. The 8088 supports 16-bit operations, including multiplication and division, and supports 20 bits of addressing (1 megabyte of storage). It operates in the minimum mode at 4.77 MHz. This frequency, which is derived from a 14.31818-MHz crystal, is divided by 3 for the processor clock, and by 4 to obtain the 3.58-MHz color-burst signal required for color televisions.

For additional information about the 8088, refer to the publications listed in "Bibliography".

The processor is supported by a set of high-function support-devices providing three 16-bit timer-counter channels, and nine prioritized-interrupt levels.

The three programmable timer/counters are provided by an Intel 8253-5 programmable interval-timer and are used by the system in the following manner: Channel 0 is used as a general-purpose timer providing a constant time-base for implementing a time-of-day clock; Channel 1 is used to deserialize the keyboard data and for time-of-day overflow during diskette operations. Channel 2 is used to support the tone generation for the audio speaker and to write data to the cassette.

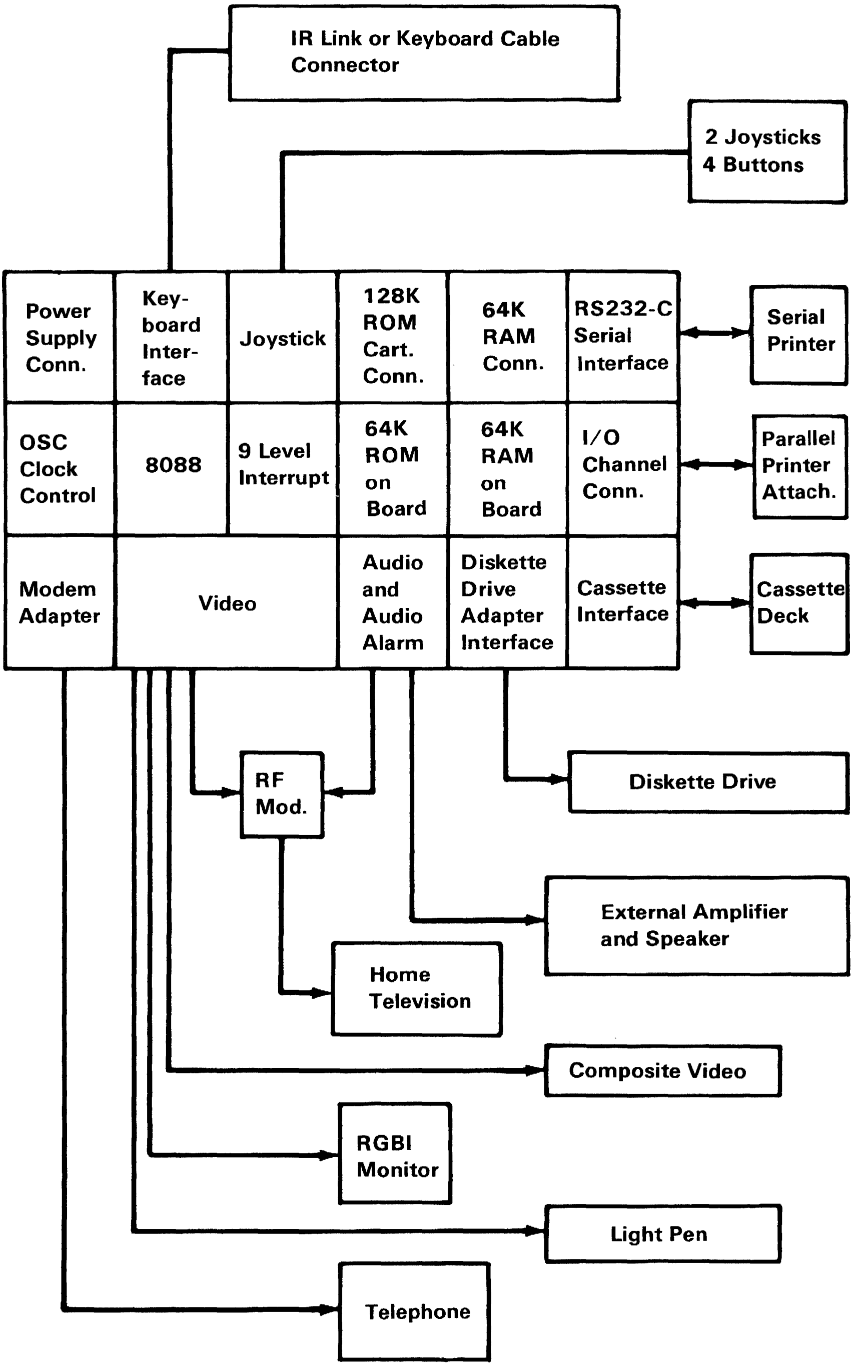
Of the nine prioritized levels of interrupt, three are bused to the system's I/O channel for use by adapters. Five levels are used on the system board. Level 0, the highest priority, is attached to Channel 0 of the timer/counter and provides a periodic interrupt for the time-of-day clock; level 3 is the serial-port-access interrupt; level 4 is the modem-access interrupt; level 5 is the vertical-retrace interrupt for the video; and level six is the diskette drive adapter-access interrupt. The non-maskable interrupt (NMI) of the 8088 is attached to the keyboard-interface circuits and receives an interrupt for each scan code sent by the keyboard.

The system board supports both read-only memory (ROM) and R/W memory (RAM). It has space for 64K bytes by 8 bits of ROM. There are two module sockets that accept a 32K byte by 8 bit ROM module. ROM is aligned at the top of the 8088's address space. This ROM contains the Power-On Self-Test, cassette-BASIC interpreter, cassette-operating system, I/O drivers, dot patterns for 256 characters in graphics mode, a diskette bootstrap-loader and user-selectable diagnostic-routines.

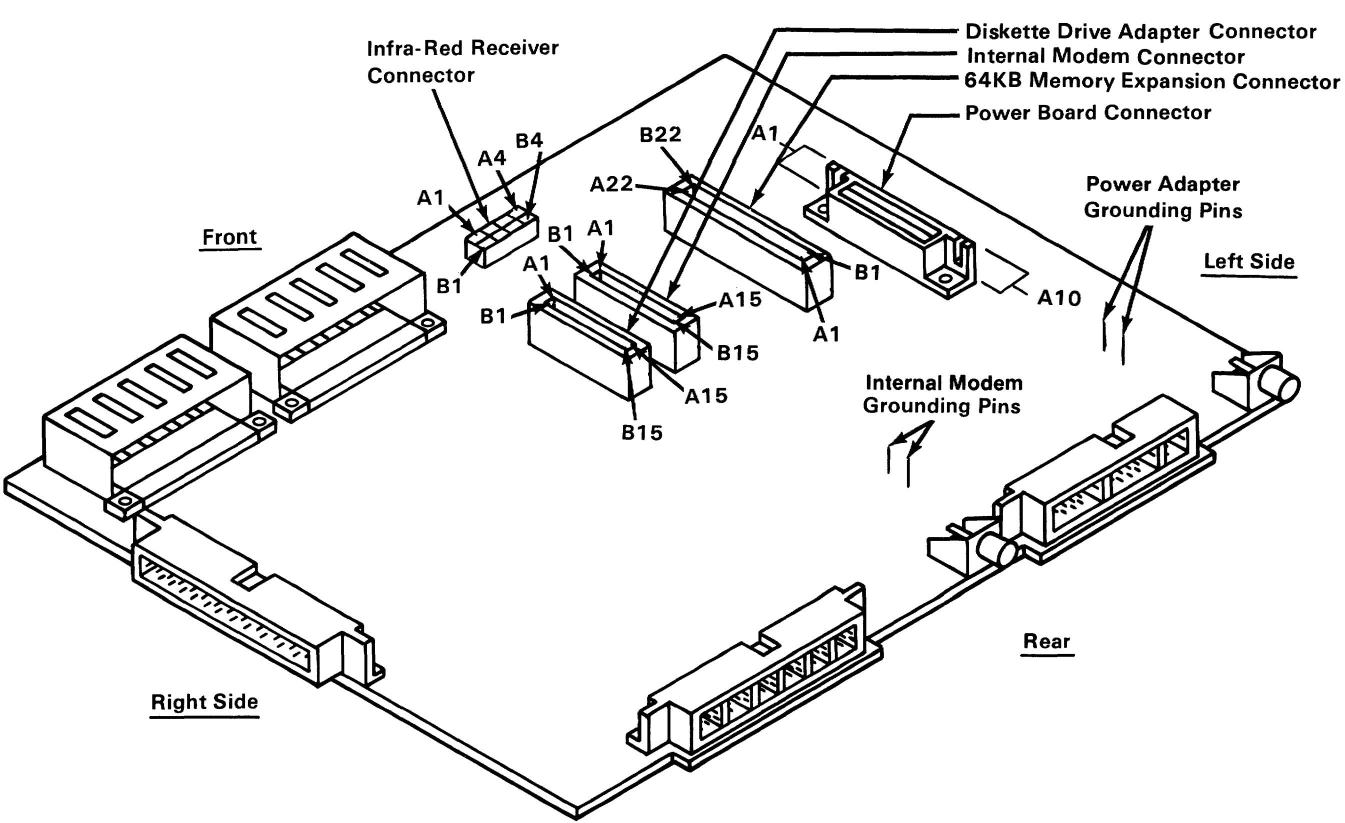
The system board contains the following major functional components:

- 8088 Microprocessor
- 64K ROM
- 128K ROM Cartridge Interface
- 64K Dynamic RAM
- 64KB Memory and Display Expansion Interface
- Serial Port (RS232)
- Audio Alarm (Beeper)
- Sound Subsystem
- Cassette Interface
- Joystick Interface
- Keyboard Interface
- Modem Interface
- Diskette Interface
- Video/Graphics Subsystem
- Light Pen Interface
- I/O Expansion Bus
- 9-Level Interrupt

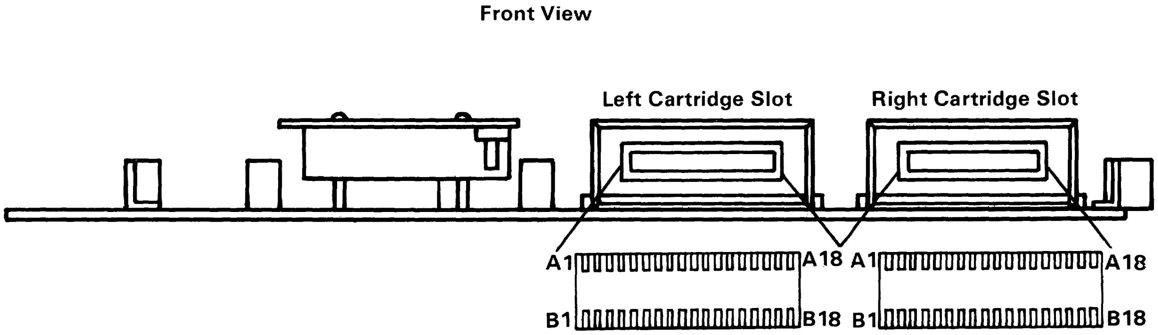
The following is a block diagram of the System Board.

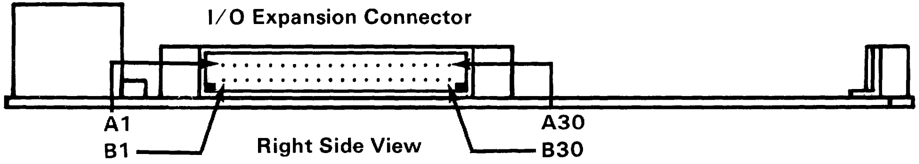


System Board Block Diagram


Infra-Red Receiver Connector
Diskette Drive Adapter Connector
Internal Modem Connector
64KB Memory Expansion Connector
Power Board Connector
B4
A4
A1
B1
B22
A22
A1
B1
A1
A10
B1
A1
A15
B15
A1
B1
A15
B15
Front
Power Adapter Grounding Pins
Left Side
Internal Modem Grounding Pins
Rear
Right Side

System Board Connector Specifications (Part 1 of 3)

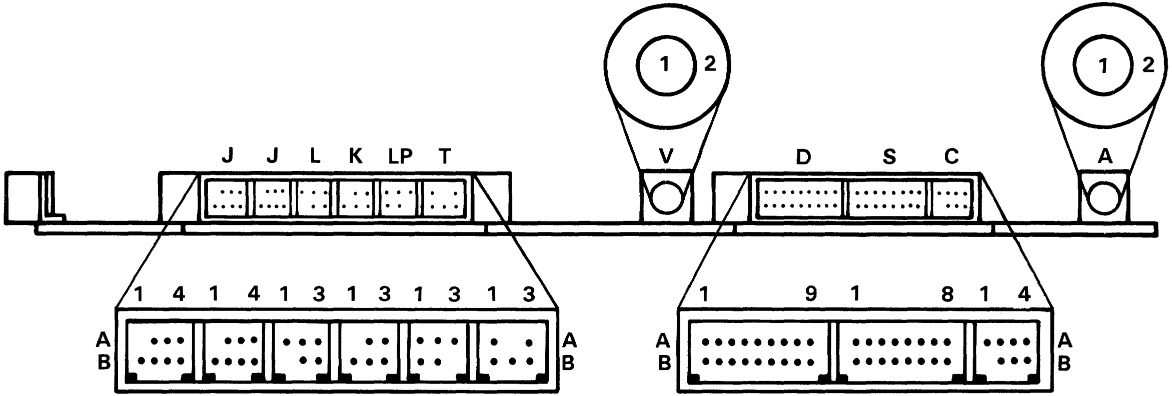




System Board Connector Specifications (Part 2 of 3)

Letter Designation	Connector Use
J	Left Joystick
J	Right Joystick
L	Spare
K	Keyboard
LP	Light Pen
T	Television

Letter Designation	Connector Use
V	Composite Video
D	Direct Drive Video
S	Serial Device
C	Cassette
A	Audio



System Board Connector Specifications (Part 3 of 3)

Processor and Support

The (R) Intel 8088 Microprocessor is used as the system's central processor. Some of its characteristics are:

- 4.77 MHz clock
- 20 bit address bus
- 8-bit memory interface
- 16-bit ALU (arithmatic/logic unit) and registers
- Extensive instruction set
- DMA and interrupt capabilities
- Hardware fixed-point multiply and divide

The system clock is provided by one Intel 8284A clock chip. The 8088 is operated in the minimum mode.

Performance

The 8088 is operated at 4.77 MHz which results in a clock cycle-time of 210 ns.

Normally four clock cycles are required for a bus cycle so that an 840 ns ROM memory cycle time is achieved. RAM write and read cycles will incur an average of two wait states because of sharing with video, leading to an average of six clock cycles. I/O reads and writes also take six clock cycles leading to a bus cycle time of 1.260 μs.

Notes:

8259A Interrupt Controller

PC*jr* Hardware Interrupts

Nine hardware levels of interrupts are available for the PC*jr* system. The highest-priority interrupt is the NMI interrupt in the 8088. The NMI is followed by eight prioritized interrupt-levels (0-7) in the 8259A Programmable Interrupt Controller, with IRQ 0 as the highest and IRQ 7 as the lowest. The interrupt level assignments follow:

Level		Function
8088	NMI	Keyboard Interrupt
8259A	IRQ 0	Timer Clock Interrupt
8259A	IRQ 1	I/O Channel (Reserved)
8259A	IRQ 2	I/O Channel
8259A	IRQ 3	Asynchronous Port Interrupt (RS-232C)
8259A	IRQ 4	Modem Interrupt
8259A	IRQ 5	Vertical Retrace Interrupt (Display)
8259A	IRQ 6	Diskette Interrupt
8259A	IRQ 7	I/O Channel (Parallel Printer)

Hardware Interrupts

8259A Programming Considerations

The 8259A is set up with the following characteristics:

- Buffered Mode
- 8086 Mode
- Edge Triggered Mode
- Single Mode Master (No Cascading is Allowed)

The 8259A I/O is located at I/O address hex 20 and hex 21. The 8259A is set up to issue interrupt types hex 8 to hex F which use pointers to point to memory address hex 20 to hex 3F.

The following figure is an example setup.

0263 BO 13	MOV AL, 13H	; ICW1 - Reset edge sense circuit set single ; 8259 Chip and ICW4 read
0265 E6 20	OUT INTA00,AL	
0267 BO 08	MOV AL,8	; ICW2 - Set interrupt type 8 (8-F)
0269 E6 21	OUT INTA01,AL	
026B BO 09	MOV AL,9	; ICW4 - Set buffered mode/master and 8086 mode
026D E6 21	OUT INTA01,AL	

Example Set Up

64K RAM

The 64K bytes of R/W memory reside on the system board and require no user configuration.

Eight 64K byte by 1, 150 ns, dynamic memory modules are used to provide 64K byte of storage. The RAM has no parity. Sources of these memory modules include the Motorola MCM6665AL15 and the Texas Instruments TMS4164-15 or equivalent.

The system board 64K RAM is mapped at the bottom of the 1 MEG address space. The system board 64K RAM is mapped to the next 64K bytes of address space if the 64KB Memory and Display Expansion option is not installed. If read or written to, this higher block of address space will look just like the low-order 64K-byte block. This means the bottom 128K bytes of address space is always reserved for RAM. If the 64KB Memory and Display Expansion option is installed, it is mapped to the 'ODD' memory space within the 128K byte-reserved space while the system board memory is mapped to the 'EVEN' space. Memory refresh is provided by the 6845 CRT Controller and gate array. The gate array cycles the RAM and resolves contention between the CRT and processor cycles.

See "IBM PC*jr* 64KB Memory and Display Expansion" in Section 3 for a detailed description.

Notes:

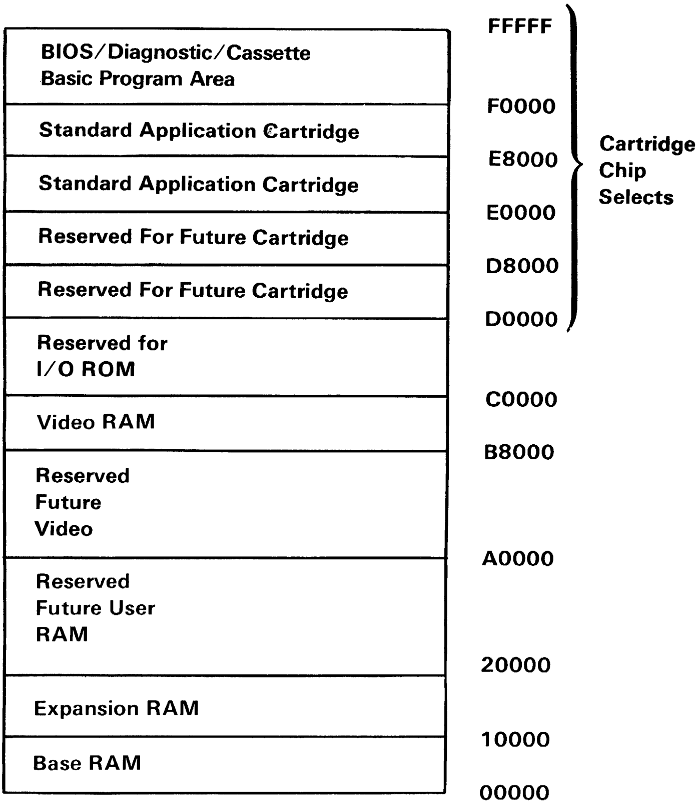
ROM Subsystem

The ROM subsystem is made up of 64K bytes of ROM aligned at the top of the 1 MEG address space. The ROM is built using 32K byte by 8 ROM-modules. The ROM has no parity. The general memory specifications for the ROM are:

```
Access Time   -  250 ns
Cycle Time    -  375 ns
```

ROM modules Mk 38000 from Mostek, TMM23256P or equivelent are used. Address A14 is wired to both pin 1 and pin 27.

The following figure is a map of the sections of memory allocated for use by the system:


BIOS/Diagnostic/Cassette
Basic Program Area
Standard Application Cartridge
Standard Application Cartridge
Reserved For Future Cartridge
Reserved For Future Cartridge
Reserved for
I/O ROM
Video RAM
Reserved
Future
Video
Reserved
Future User
RAM
Expansion RAM
Base RAM
FFFFF
F0000
E8000
E0000
D8000
D0000
C0000
B8000
A0000
20000
10000
00000
Cartridge
Chip
Selects

Memory Map

Input Output Channel

The Input/Out channel (I/O) is an extension of the 8088 microprocessor bus. It is however, demultiplexed, repowered, and enhanced by the addition of interrupts.

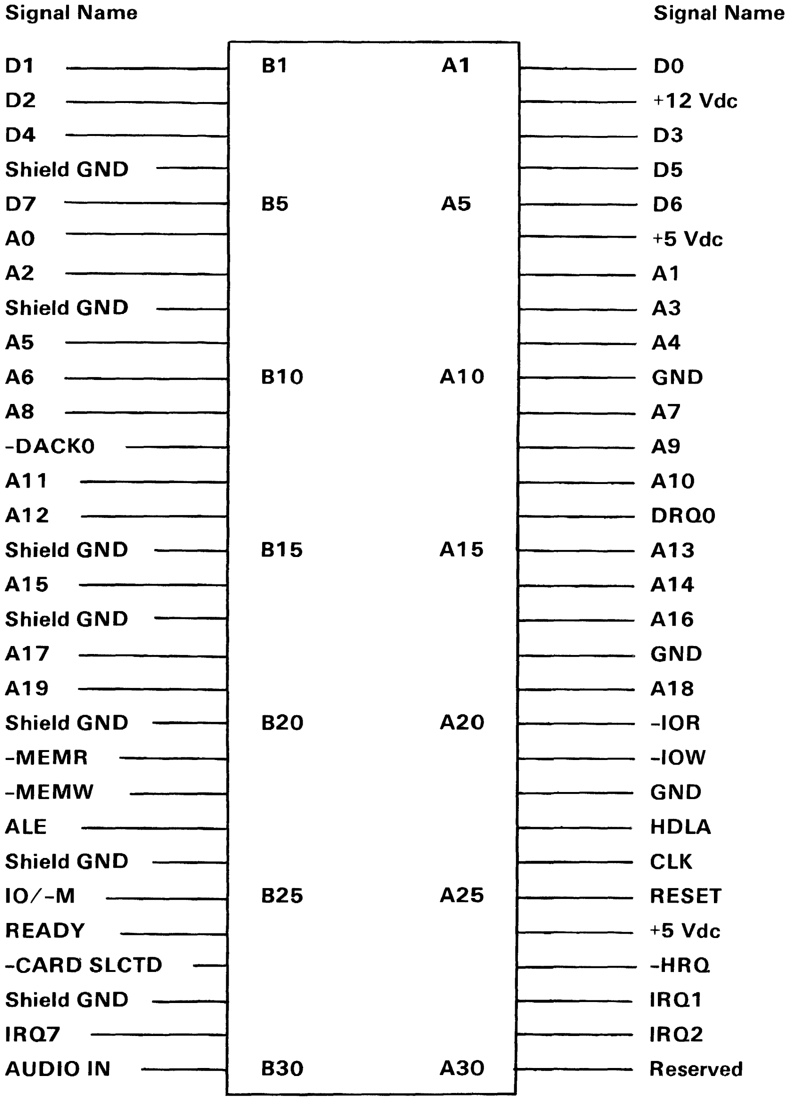
The I/O channel contains an 8-bit bidirectional bus, 20 address lines, 3 levels of interrupt, control lines for memory and I/O read or write, clock and timing lines, and power and ground for the adapters. Voltages of +5 dc and +12 dc are provided for external adapters. Any additional power needs will require a separate power-module.

All I/O Channel functions are bused to the right-hand side of the system unit and are provided by a right-angle, 60-pin connector. Each external adapter connects to the I/O bus and passes the bus along for the next attachment.

A 'ready' line is available on the I/O Channel to allow operation with slow I/O or memory devices. If the channel's 'ready' line is not activated by an addressed device, all processor-generated memory-read and write cycles take four 210-ns clocks or 840-ns/byte. All processor-generated I/O-read or write cycles require six clocks for a cycle time of 1.26-μs/byte.

The I/O Channel also contains the capability to add bus masters to the channel. These devices could be DMA devices or alternate processors.

The I/O Channel signals have sufficient drive to support five I/O Channel expansion-adapters and the internal modem and diskette drive adapter, assuming one standard TTL load per attachment. For information on power available for external adapters, see "System Power Supply", later in this Section.



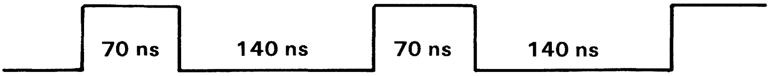
Signal Name			Signal Name
D1	B1	A1	D0
D2			+12 Vdc
D4			D3
Shield GND			D5
D7	B5	A5	D6
A0			+5 Vdc
A2			A1
Shield GND			A3
A5			A4
A6	B10	A10	GND
A8			A7
-DACK0			A9
A11			A10
A12			DRQ0
Shield GND	B15	A15	A13
A15			A14
Shield GND			A16
A17			GND
A19			A18
Shield GND	B20	A20	-IOR
-MEMR			-IOW
-MEMW			GND
ALE			HDLA
Shield GND			CLK
IO/-M	B25	A25	RESET
READY			+5 Vdc
-CARD SLCTD			-HRQ
Shield GND			IRQ1
IRQ7			IRQ2
AUDIO IN	B30	A30	Reserved

I/O Channel Expansion Connector Specifications

System Board I/O Channel Description

The following is a description of the I/O Channel. All signals are TTL compatible.

Signal	I/O	Description
CLK	O	System Clock: It is a divide-by-three of the 14.31818 MHz oscillator and has a period of 210 ns (4.77 MHz). The clock has a 33% duty cycle.



Duty Cycle

Signal	I/O	Description
RESET	O	This line is used to reset or initialize system logic upon power-up. This line is synchronized to the falling edge of the clock and is 'active high'. Its duration upon power up is 26.5 μs.
A0-A19	I/O	Address Bits 0 to 19: These lines are used to address memory and I/O devices within the system. The 20 address lines allow access of up to 1 megabyte of memory. A0 is the least-significant- bit (LSB) while A19 is the most-significant-bit (MSB). These lines are normally driven by the 8088 microprocessor as

outputs, but can become inputs from an external bus-master by issuing an HRQ and receiving an HLDA.

D0-D7 I/O Data Bits 0-7: These lines provide data-bus bits 0 to 7 for the processor, memory, and I/O devices. D0 is the least-significant-bit (LSB) and D7 is the most-significant-bit (MSB). These lines can be controlled by an external bus-master by issuing an HRQ and receiving an HLDA.

ALE O Address Latch Enable: This line is provided to allow the addition of wait states in memory and I/O cycles.

READY I This line, normally 'high' ('ready'), is pulled 'low' ('not ready') by a memory or I/O device to lengthen I/O or memory cycles. It allows slower devices to attach to the I/O Channel with a minimum of difficulty. Any slow device requiring this line should drive it 'low' immediately upon detecting a valid address and IO/-M signal. Machine cycles (I/O and memory) are extended by an integral number of CLK cycles (210 ns). Any bus master on the I/O Channel should also honor this 'ready' line. It is pulled 'low' by the system board

		on memory read and write cycles and outputting to the sound subsystem.
IRQ1, IRQ2, IRQ7	I	Interrupt Request 1, 2, and 7: These lines are used to signal the processor that an I/O device requires attention. They are prioritized with IRQ1 as the highest priority and IRQ7 as the lowest. An Interrupt Request is generated by raising an IRQ line ('low' to 'high') and holding it 'high' until it is acknowledged by the processor (interrupt-service routine).
-IOR	I/O	I/O Read Command: This command line instructs an I/O device to drive its data onto the data bus. This signal may be driven by the 8088 microprocessor or by an external bus-master after it has gained control of the bus. This line is active 'low'.
-IOW	I/O	I/O Write Command: This command line instructs an I/O device to read the data on the data bus. This signal may be driven by the 8088 microprocessor or by an external bus-master after it has gained control of the bus. This line is active 'low'.
-MEMR	I/O	Memory Read Command: This command line instructs the

		memory to drive its data onto the data bus. This signal may be driven by the 8088 microprocessor or by an external bus-master after it has gained control of the bus. This line is active 'low'.
-MEMW	I/O	Memory Write Command: This command line instructs the memory to store the data present on the data bus. This signal may be driven by the 8088 microprocessor or by an external bus-master after it has gained control of the bus. This line is active low.
IO/-M	I/O	I/O or Memory Status: This status line is used to distinguish a memory access from an I/O access. This line should be driven by a bus master after it has gained control of the bus. If this line is 'high' it indicates an I/O Address is on the Address Bus; if this line is 'low', it indicates a memory address is on the Address Bus.
-HRQ	I	Hold Request: This line indicates that another bus master is requesting the I/O Channel. To gain bus-master status, a device on the channel must assert -HRQ (active 'low'). The 8088 will respond to a -HRQ by asserting an HLDA. After receiving an HLDA, the new bus master may

		control the bus, and must continue to assert the -HRQ until it is ready to relinquish the bus. A -HRQ is not an asynchronous signal and should be synchronized to the system clock. All channel devices with bus-master capabilities must latch data-bit D4 during any 'Out' instruction to A0-A7. The resulting signal should be used to qualify -HRQ as follows: Latched value = 1 --> -HRQ is inhibited. Latched value = 0 --> -HRQ is allowed. For more detail, see the explanation of the A0 port.
DRQ 0	0	This line comes from the floppy disk controller (FDC) and can be used by an external DMA to indicate that a byte should be transferred to the FDC.
-DACK 0	I	This line should come from an external DMA and should indicate that a byte is being transferred from memory to the FDC.
HLDA	O	Hold Acknowledge: This line indicates to a bus master on the channel that -HRQ has been honored and that the 8088 has floated its bus and control lines.

-CARD SLCTD I This line should be pulled down by any adapter when it is selected with address and IO/-M. This line will be used for bus expansion. It is pulled up with a resistor and should be pulled down with an open collector device.

AUDIO IN I Channel devices may provide sound sources to the system-board sound-subsystem through this line. It is 1 volt peak-to-peak, dc biased at 2.5 volts above ground.

Input/Output

Hex Range	9	8	7	6	5	4	3	2	1	0	Device
20-27	0	0	0	0	1	0	0	X	X	A0	PIC 8259
40-47	0	0	0	1	0	0	0	0	A1	A0	Timer 8253-5
60-67	0	0	0	1	1	0	0	X	A1	A0	PPI 8255-5
A0-A7	0	0	1	0	1	0	0	X	X	X	NMI Mask Reg.
C0-C7	0	0	1	1	0	0	0	X	X	X	Sound SN76496N
F0-FF	0	0	1	1	1	1	X	A2	A1	A0	Diskette
200-207	1	0	0	0	0	0	0	X	X	X	Joystick
2F8-2FF	1	0	1	1	1	1	1	A2	A1	A0	Serial Port
3D0-3DF	1	1	1	1	0	1	A3	A2	A1	A0	Video Subsystem
3F8-3FF	1	1	1	1	1	1	1	A2	A1	A0	Modem

I/O Map

X = Don't care (that is, not in decode.)

- Any I/O which is not decoded on the system board may be decoded on the I/O Channel.
- At Power-On time the NMI into the 8088 is masked 'off'. This mask bit can be set by system software as follows:

Write to Port A0 D7=ENA NMI D6=IR TEST ENA D5=SELC CLK1 INPUT D4=+Disable HRQ

8255 Bit Assignments

PA	Output
PA0	Reserved for Keystroke Storage
PA1	Reserved for Keystroke Storage
PA2	Reserved for Keystroke Storage
PA3	Reserved for Keystroke Storage
PA4	Reserved for Keystroke Storage
PA5	Reserved for Keystroke Storage
PA6	Reserved for Keystroke Storage
PA7	Reserved for Keystroke Storage
PB	Output
PB0	+Timer2 Gate (Speaker)
PB1	+Speaker Data
PB2	+Alpha (-Graphics)
PB3	+Cassette Motor Off
PB4	+Disable Internal Beeper and Cassette Motor Relay
PB5	SPKR Switch 0
PB6	SPKR Switch 1
PB7	Reserved
PC	Input
PC0	Keyboard Latched
PC1	-Internal MODEM Card Installed
PC2	-Diskette Drive Card Installed
PC3	-64KB Memory and Display Expansion Installed
PC4	Cassette Data In
PC5	Timer Channel 2 Output
PC6	+Keyboard Data
PC7	-Keyboard Cable Connected

8255 Bit Assignment Description

PA0 thru PA7 (Output Lines) Port A is configured as an output. The output lines are not used by the hardware, but are used to store keystrokes. This is done to maintain compatibility with the Personal Computer, and Personal Computer XT.

PB0 (+Timer 2 Gate) This line is routed to the gate input of timer 2 on the 8253-5. When this bit is 'low', the counter operation is halted. This bit and PB1 (+Speaker Data) controls the operation of the 8253-5 sound source.

PB1 (+Speaker Data) This bit ANDS 'off' the output of the 8253-5 timer 2. It can be used to disable the 8253-5 sound source, or modify its output. When this bit is a 1, it enables the output, a 0 forces the output to zero.

PB2 (+Alpha -Graphics) This bit is used to steer data from the memory into the Video Gate Array. This bit should be a 1 for all alpha modes, and a 0 for all graphics modes.

PB3 (+Cassette Motor Off) When this bit is a 1, the cassette relay is 'open' and the cassette motor is 'off'. When this bit is a 0, and PB4 = 0, the cassette motor is 'on'.

PB4 (+Disable internal beeper and cassette motor relay) When this bit is a 1, the internal beeper is 'disabled' and the 8253-5 timer 2 sound source can only be heard if it is steered to the audio output. This bit also disables the cassette motor when it is a 1. To 'enable' the cassette motor, this bit must be a 0. In this case, PB1 should be used to gate 'off' the internal beeper and 8253-5 sound source.

PB5, PB6 (Speaker switch 0,1) These bits steer one of 4 sound sources. This is available to the RF modulator or the external audio jack. The sound sources selected are shown below.

PB6	**PB5**	**Sound Source**
0	0	8253-5 Timer 2
0	1	Cassette Audio Input
1	0	I/O Channel Audio In
1	1	76496

PB7 (Open) Reserved for future use.

PC0 (Keyboard latched) This input comes from a latch which is set to a 1 on the first rising edge of the Keyboard Data stream. The output of this latch also causes the NMI to occur. This latch is cleared by doing a dummy 'Read' operation to port A0. This input is provided so that a program can tell if a keystroke occurred during a time when the NMI was masked 'off' and a keystroke has been missed. The program will then be able to give an error indication of the missed keystroke.

PC1 (-Modem card installed) When this bit is a 0, it indicates that the Internal Modem card is installed.

PC2 (-Diskette card installed) When this bit is a zero, it indicates that the Diskette Drive Adapter is installed.

PC3 (-64KB Memory and Display Expansion installed) When this bit is a 0, it indicates that the 64KB Memory and Display Expansion is installed.

PC4	(Cassette data in)	If the cassette-motor relay is 'closed', and the cassette motor is 'on', this pin will contain data which has been wave shaped from the cassette. If the cassette-motor relay is 'off', this pin will contain the same data as the 8253-5 timer 2 output.
PC5	(Timer channel 2 output)	This input is wired to the timer channel 2 output of the 8253-5.
PC6	(+Keyboard data)	This input contains keyboard data. The keyboard data comes from the cable if attached, or from the IR Receiver if the cable is not attached.
PC7	(-Keyboard cable connected)	If this bit is 'low', it indicates that the keyboard cable is connected.

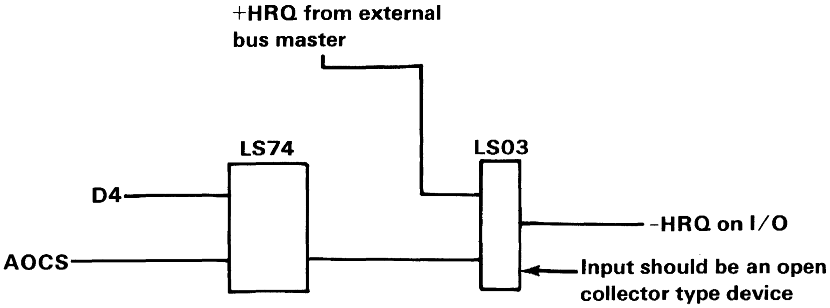
Port A0 Output Description

D7 **(Enable NMI)** When this bit is a 1, the NMI is 'enabled'. When it is a 0, it is 'disabled'.

D6 **(IR test ENA)** This bit enables the 8253-5 timer 2 output into an IR diode on the IR Receiver board. This information is then wrapped back to the keyboard input. If the cable is not connected, timer 2 should be set for 40 kHz which is the IR-modulation frequency. This feature is used only for a diagnostic test of the IR Receiver board.

D5 **(Selc Clk1 input)** This bit selects one of two input Clks to the 8253-5 timer 1. A 0 selects a 1.1925 MHz Clk input used to`assist the program in de-serializing the keyboard data. A 1 selects the timer 0 output to be used as the Clk input to timer 1. This is used to catch timer 0 overflows during diskette drive operations when interrupts are masked 'off'. This is then used to update the time-of-day.

D4 **(+Disable HRQ)** This bit is not actually implemented on the system board, but is supported by the programming. This bit is used to disable -HRQs from external bus-masters (DMA, Alternate Processors, etc.) The logic for this bit must exist on each bus-master attachment. A 0 should 'enable' -HRQ, and a 1 should 'disable' -HRQ.



Port A0 Output Description

Port A0 Input Operation

A 'read' to I/O port A0 will clear the keyboard NMI latch. This latch causes an NMI on the first rising edge of the keyboard data if the enable NMI bit (port A0 bit D7) is 'on'. This latch can also be read on the 8255 PC0. The program can determine if a keystroke occurred while the NMI was 'disabled' by reading the status of this latch. This latch must be cleared before another NMI can be received.

The System board provides for selection of keyboard data from either a cable or the IR-receiver board. The IR-receiver board is mounted on the system board and can receive data through an IR link. The source of the keyboard's data is determined by the -Cable Connected signal at the keyboard cable connector. Keyboard serial data is available to the 8088 at bit PC6 of the 8255 PPI.

The system board is responsible for the de-serialization of keyboard data. The start bit in the serial stream causes an NMI to be generated. The 8088 then reads the 8253 timer to determine when to interrogate the serial stream. After de-serialization the NMI service-routine does a 'Read' from hex A0 to clear the NMI latch.

During certain time-critical operations, such as diskette I/O, the processor will mask 'off' the NMI interrupt. Keyboard inputs during this time cannot be serviced. A keyboard latch is provided so that at the end of such operations the processor will determine whether any keys were pressed and take appropriate actions. The keyboard latch is 'set' by any key being pressed and is 'reset' by 'Reading' the NMI port. (No data is presented to the microprocessor during this 'Read'.) Keyboard latch data is available to the processor at bit PC0 of the 8255 PPI.

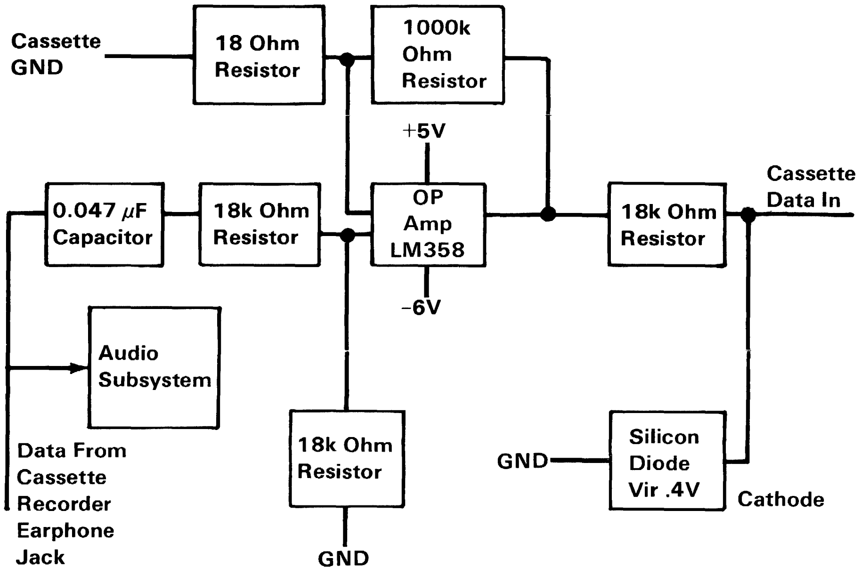
Notes:

Cassette Interface

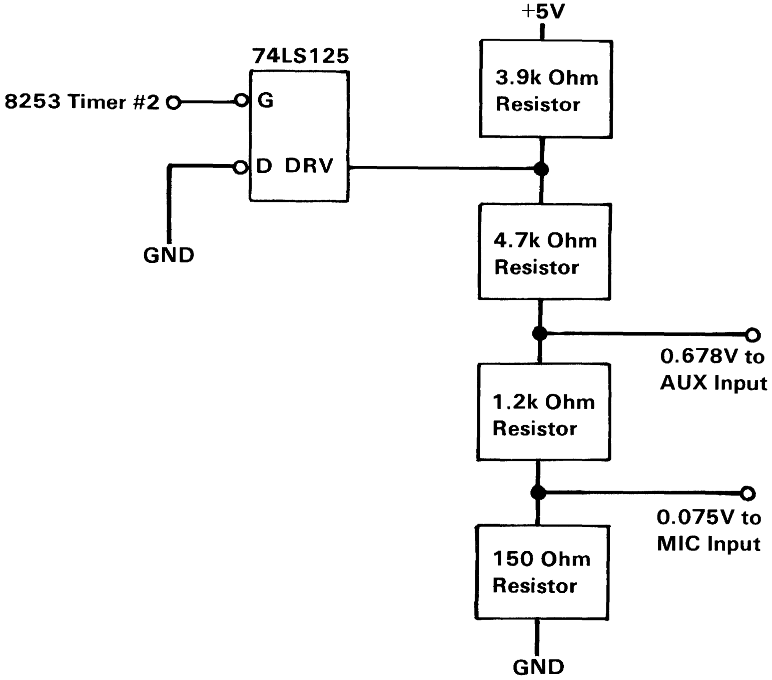
The cassette interface is controlled through software. An output from the 8253 timer controls the data to the cassette recorder through the cassette connector at the rear of the system board. The cassette-input data is read by an input-port bit of the 8255A-5 programmable-peripheral-interface (PPI) (8255A-5 PC4). Software algorithms are used to generate and read cassette-data. The cassette drive- motor is controlled by Bit PB3 of the 8255. Bit PB4, which 'enables' the 7547 relay driver, must be 'low' when the motor is to be turned on. The cassette interface has a wrap feature which connects the output to the input when the motor control is 'off'. See "BIOS Cassette Logic" in Section 5 for information on data storage and retrival.

A mechanism is provided that will direct the cassette input to the audio subsystem. Please see "Sound Subsection" in Section 2.

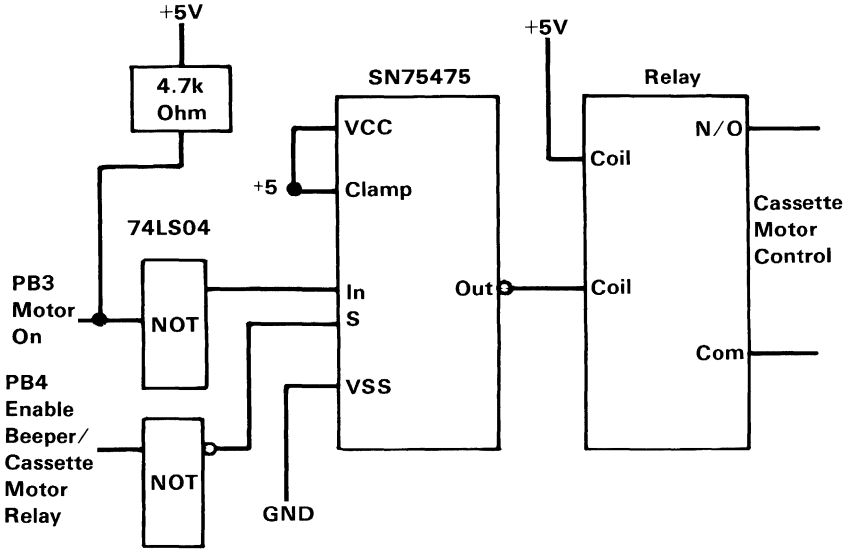
Circuit block diagrams for the cassette-interface read, write, and motor control are illustrated in the following figures.



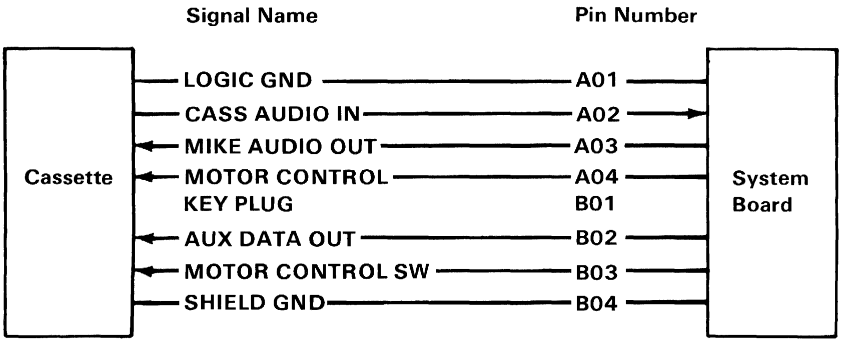
Cassette-Interface Read-Hardware Block Diagram



Cassette-Interface Write-Hardware Block Diagram



Cassette-Motor Control Block Diagram



Signal Name	Pin Number
LOGIC GND	A01
CASS AUDIO IN	A02
MIKE AUDIO OUT	A03
MOTOR CONTROL	A04
KEY PLUG	B01
AUX DATA OUT	B02
MOTOR CONTROL SW	B03
SHIELD GND	B04

Cassette Connector Specifications

Notes:

Video Color/Graphics Subsystem

The video subsystem is designed so that the IBM Color Display, composite monitors, and a home television set can be attached. It is capable of operating in black-and-white or color. It provides three video ports: a composite-video, a direct-drive, and a connector for an RF modulator to be used with home televisions. In addition, it contains a light pen interface.

Note: The IBM Personal Computer Monochrome Display cannot be used with the PC*jr* system.

Note: An IBM Connector for Television option must be obtained to attach a home TV.

The subsystem has two basic modes of operation: alphanumeric (A/N) and all points addressable graphics (APA). Additional modes are available within the A/N and APA modes.

In the A/N mode, the display can be operated in either a 40-column by 25-row mode for a low-resolution display home television, or an 80-column by 25-row mode for high-resolution monitors. In both modes, characters are defined in an 8-wide by 8-high character box and are 7-wide by 7-high, with one line of descender. Both A/N modes can operate in either color or black-and-white.

In the A/N black-and-white mode, the character attributes of reverse video, blinking, highlighting and gray shades are available.

In the A/N color mode, sixteen foreground-colors and sixteen background-colors are available for each character. In addition, blinking on a per-character basis is available. When blinking is used, only eight background-colors are available. One of 16 colors, or gray shades can be selected for the screen's border in all A/N modes.

In both A/N modes, characters are formed from a ROM character-generator. The character generator contains dot patterns for 256 different characters. The character set contains the following major groupings of characters:

- 16 special characters for game support
- 15 characters for word-processing editing support
- 96 characters for the standard-ASCII-graphics set
- 48 characters for foreign-language support
- 48 characters for business block-graphics (allowing drawing of charts, boxes, and tables using single or double lines)
- 16 selected Greek symbols
- 15 selected scientific-notation characters

In the APA mode, there are three resolutions available: a low-resolution mode (160 PELs [Picture ELements] by 200 rows), a medium-resolution mode (320 PELs by 200 rows), and a high-resolution mode (640 PELs by 200 rows).

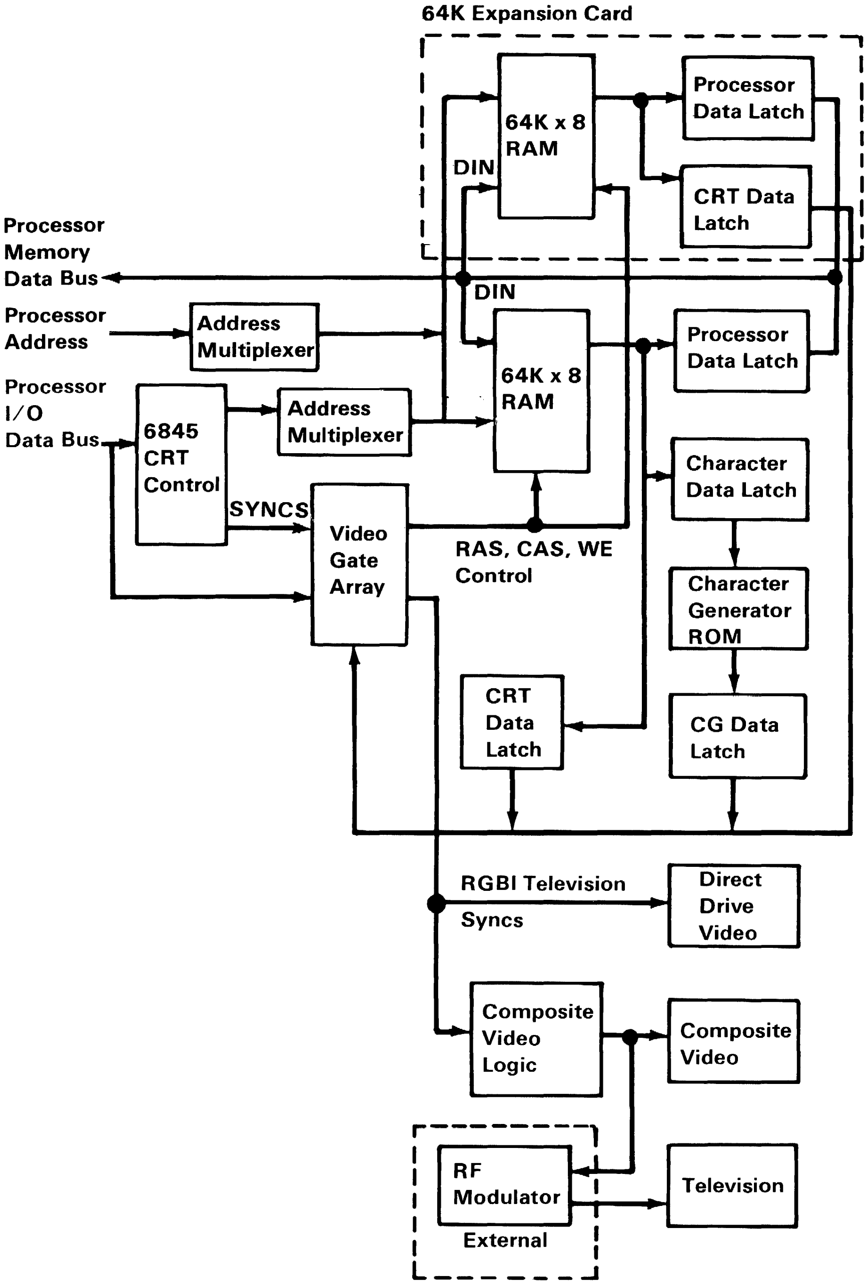
Different color modes exist within each of the APA resolutions. Two, four, or sixteen colors are available in APA color, and two, four, or sixteen gray shades are available in APA black-and-white.

One of sixteen colors, or grey shades can be selected for the screen's border in all APA modes.

The direct drive, composite video and RF Modulator connector are right-angle-mounted connectors extending through the rear of the system unit.

The video color/graphics subsystem is implemented using a Motorola 6845 CRT controller device and a Video Gate Array (VGA) (LSI5220). The video subsystem is highly programmable with respect to raster and character parameters. Thus many additional modes are possible with the proper programming.

The following figure shows a block diagram of the video color/graphics subsystem.



Video Color/Graphic Subsystem Block Diagram

Major Components Definitions

Motorola 6845 CRT Controller

This device provides the necessary interface to drive a raster-scan CRT. Additional information about this component is provided in publications listed in "Bibliography".

Storage Organization

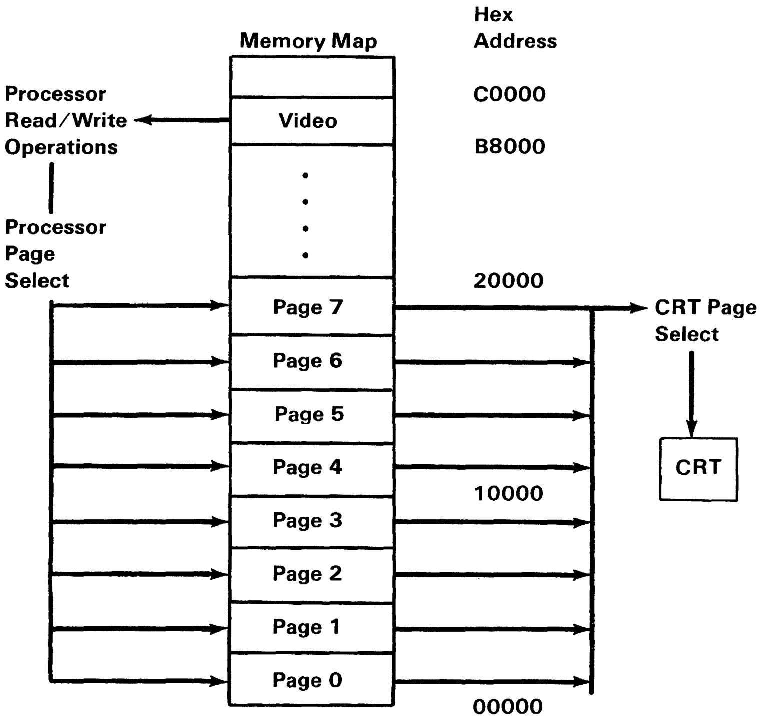
The base video-color/graphics-subsystem accesses 64K bytes of read/write memory (RAM). A 64KB Memory and Display Expansion can be added to increase the amount of system RAM to 128K bytes. This memory-storage area serves two functions; as the video-display buffer and as the system processor is (8088) main-RAM.

The RAM is located at address hex 0000 and is either 64K bytes or 128K bytes with the memory expansion option. The 8088 can access the memory by reading from and writing to address locations hex 00000 to 1FFFF or by reading from or writing to the 16K-byte region starting at address hex B8000. The page affected by a read or write operation is determined by the processor's page register. The processor can access the RAM at any time in all modes with no adverse effect to the video information. The page that the video information is taken from is determined by the CRT page register.

The processor and CRT page registers are write only registers and can be changed at any time. These registers allow the processor to work in one page while the display is displaying another page. The processor can switch pages at the vertical-retrace time. This will aid animation on the video color/graphics subsystem.

Also, since all 128K bytes of read/write memory are available for display purposes, the application can use as little or as much memory as needed for the display.

The following figure is a map of the video color/graphics subsystem.



Video Color/Graphics Subsystem Memory Map

Bandwidth

The video bandwidth is either 3.5, 7 or 14 MHz depending on the mode of operation. The processor bandwidth is the same for all modes. The processor is allowed one cycle every 1.1 microseconds. An average of two wait states will be inserted in a processor RAM read cycle, because the average latency time for the processor to get a cycle is 560 ns and the cycle time is 350 ns. There is no performance penalty for redirecting processor reads and writes through the B8000 - BFFFF address area.

Character Generator

The ROM character-generator consists of 2K bytes of storage which cannot be read from, or written to under software control. It is implemented with a MCM68A316E or equivalent. Its specifications are 350 ns access, 350 ns cycle static operation. The device is pin compatible with 2716 and 2732 EPROMS.

Video Gate Array

A CMOS gate array is used to generate storage-timing (RAS, CAS, WE), direct-drive, composite-color and status signals. See “Video Gate Array” later in this section.

Palette

The video color/graphics subsystem contains a 16-word by 4-bit palette in the Video Gate Array which takes PEL (Picture ELement) information from the read/write memory and uses it to select the color to display. This palette is used in all A/N and APA modes. Any input to the palette can be individually masked 'off' if a mode does not support the full complement of 16 colors. This masking allows the user to select a unique palette of colors whenever any mode does not support all 16 colors.

In two-color modes, the palette is defined by using one bit (PA0), with the following logic:

Palette Address Bit	
PA0	**Function**
0 1	Palette Register 0 Palette Register 1

Palette Logic (1 of 3)

In four-color modes, the palette is defined by using two bits (PA1 and PA0), with the following logic:

Palette Address Bits		
PA1	**PA0**	**Function**
0	0	Palette Register 0
0	1	Palette Register 1
1	0	Palette Register 2
1	1	Palette Register 3

Palette Logic (2 of 3)

In sixteen-color modes, the palette is defined by using four bits (PA3, PA2, PA1, and PA0), with the following logic:

Palette Address Bits				Function
PA3	**PA2**	**PA1**	**PA0**	
0	0	0	0	Palette Register 0
0	0	0	1	Palette Register 1
0	0	1	0	Palette Register 2
0	0	1	1	Palette Register 3
0	1	0	0	Palette Register 4
0	1	0	1	Palette Register 5
0	1	1	0	Palette Register 6
0	1	1	1	Palette Register 7
1	0	0	0	Palette Register 8
1	0	0	1	Palette Register 9
1	0	1	0	Palette Register 10
1	0	1	1	Palette Register 11
1	1	0	0	Palette Register 12
1	1	0	1	Palette Register 13
1	1	1	0	Palette Register 14
1	1	1	1	Palette Register 15

Palette Logic (3 of 3)

The sixteen colors available to all A/N and APA modes are selected through combinations of the I (Intensity), R (Red), G (Green), and B (Blue) bits. These colors are listed in the following figure:

I	R	G	B	Color
0	0	0	0	Black
0	0	0	1	Blue
0	0	1	0	Green
0	0	1	1	Cyan
0	1	0	0	Red
0	1	0	1	Magenta
0	1	1	0	Brown
0	1	1	1	Light Gray
1	0	0	0	Dark Gray
1	0	0	1	Light Blue
1	0	1	0	Light Green
1	0	1	1	Light Cyan
1	1	0	0	Pink
1	1	0	1	Light Magenta
1	1	1	0	Yellow
1	1	1	1	White

Note: The "I" bit provides extra luminance (brightness) to each available shade. This results in the light colors listed above, except for monitors that do not recognize the "I" bit.

Summary of Available Colors

Alphanumeric Modes

Every display-character position in the alphanumeric mode is defined by two bytes in the system read/write memory, using the following format:

Display Character Code Byte								Attribute Byte							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Display Format

The functions of the attribute byte are defined by the following figure:

Attribute Function	Attribute Byte Definition							
	7	6	5	4	3	2	1	0
	Fore-Ground Blink	PA2	PA1	PA0	PA3	PA2	PA1	PA0
		Background			Foreground			
Normal	B	0	0	0	I	1	1	1
Reverse Video	B	1	1	1	I	0	0	0
Nondisplay (Off)	B	0	0	0	I	0	0	0
Nondisplay (On)	B	1	1	1	I	1	1	1
I = Highlighted Foreground (Character) B = Blinking Foreground (Character)								

Attribute Functions

Graphics Mode

The Video Color/Graphics Subsystem can be programmed for a wide variety of modes within the graphics mode. Five graphics-modes are supported by the system's ROM BIOS. They are low-resolution 16-color graphics, medium-resolution 4-color graphics, medium-resolution 16-color graphics, high-resolution 2-color graphics, and high-resolution 4-color graphics. The table in the following figure summarizes the five modes:

Graphics Mode	Horiz. (PELs)	Vert. (Rows)	Number of Colors Available (Includes Background Color)
Low-Resolution 16-Color	160	200	16 (Includes b-and-w)
Medium-Resolution 4-Color	320	200	4 Colors of 16 Available
Medium-Resolution 16-Color	320	200	16 (Includes b-and-w)
High-Resolution 2-Color	640	200	2 Colors of 16 Available
High-Resolution 4-Color	640	200	4 Colors of 16 Available
Note: The screen's border color in all modes can be set to any 1 of the 16 possible colors. This border color is independent of the screen's work area colors. In Black and White each color maps to a distinct gray shade.			

Graphics Modes

Low-Resolution 16-Color Graphics

The low-resolution mode supports home-television sets, low-resolution displays, and high-resolution displays. It has the following characteristics:

- Contains a maximum of 200 rows of 160 PELs
- Specifies 1 of 16 colors for each PEL by the I, R, G, and B bits
- Requires 16K bytes of read/write memory
- Formats 2 PELs per byte for each byte in the following manner:

7	6	5	4	3	2	1	0
PA3	PA2	PA1	PA0	PA3	PA2	PA1	PA0
First Display PEL				Second Display PEL			

Low-Resolution 16-Color Graphics

Medium-Resolution 4-Color Graphics

The medium-resolution mode supports home-television sets, low-resolution displays, and high-resolution displays. It has the following characteristics:

- Contains a maximum of 200 rows of 320 PELs
- Selects one of four colors for each PEL
- Requires 16K bytes of read/write memory
- Supports 4 of 16 possible colors
- Formats 4 PELs per byte for each byte in the following manner:

7	6	5	4	3	2	1	0
PA1	PA0	PA1	PA0	PA1	PA0	PA1	PA0
First Display PEL		Second Display PEL		Third Display PEL		Fourth Display PEL	

Medium-Resolution 4-Color Graphics

Medium-Resolution 16-Color Graphics

The medium-resolution 16-color graphics mode supports home television sets, low-resolution displays, and high-resolution displays. It has the following characteristics:

- Requires system configuration of 128K bytes of read/write memory
- Requires 32K bytes of read/write memory
- Contains a maximum of 200 rows of 320 PELs.
- Specifies 1 of 16 colors for each PEL
- Formats 2 PELs per byte for each byte in the following manner.

7	6	5	4	3	2	1	0
PA3	PA2	PA1	PA0	PA3	PA2	PA1	PA0
First Display PEL				Second Display PEL			

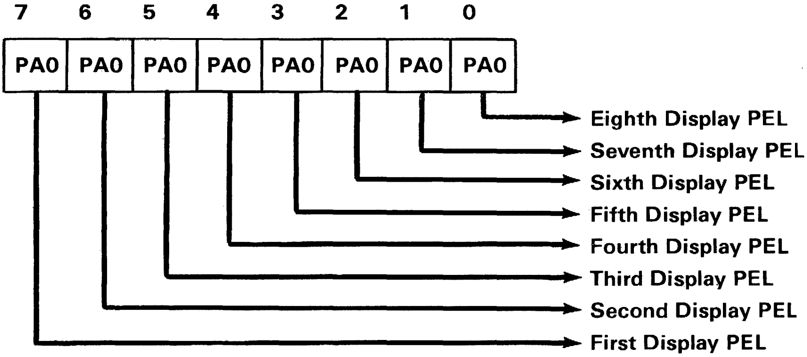
Medium-Resolution 16-Color Graphics

High-Resolution 2-Color Graphics

The high-resolution 2-color mode supports high-resolution monitors only. This mode has the following characteristics:

- Contains a maximum of 200 rows of 640 PELs
- Supports 2 of 16 possible colors.

- Requires 16K bytes of read/write memory.
- Formats 8 PELs per byte for each byte in the following manner:



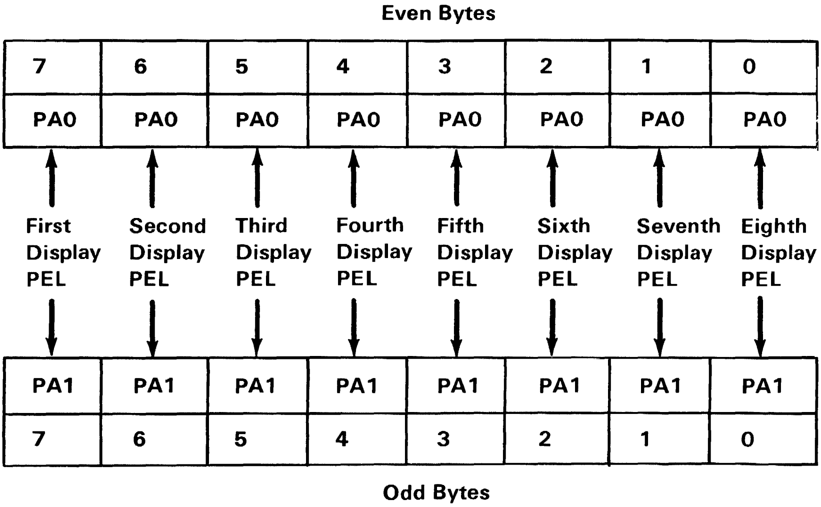
High-Resolution 2-Color Graphics

Base System

High-Resolution 4-Color Graphics

The high-resolution mode is used only with high-resolution monitors. This mode has the following characteristics:

- Requires system configuration of 128K Bytes read/write memory
- Requires 32K bytes of read/write memory
- Contains a maximum of 200 rows of 640 PELs
- Selects one of four colors for each PEL
- Supports 4 out of 16 colors
- Formats 8 PELs per two bytes (consisting of one even-byte and one odd-byte) in the following manner:

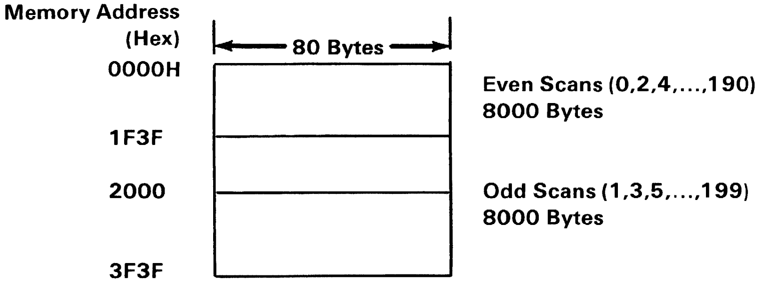


High-Resolution 4-Color Graphics

Graphics Storage Organization

For the low-resolution 16-color graphics, the medium-resolution 4-color graphics, and the high-resolution 2-color graphics, storage is organized into two banks of 8000 bytes each.

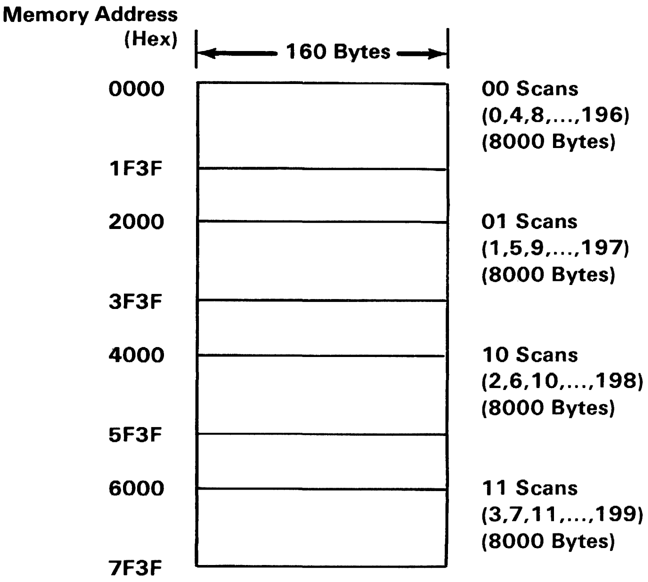
The following figure shows the organization of the graphics storage.



Graphics Storage Organization (Part 1 of 2)

Address 0000 contains PEL information for the upper-left corner of the display area.

For the medium-resolution 16-color graphics, and the high-resolution 4-color graphics modes, the graphics storage is organized into four banks of 8000 bytes each.



Graphics Storage Organization (Part 2 of 2)

Address 0000 contains PEL information for the upper-left corner of the display.

Video Gate Array

The Video Gate Array is located at I/O address hex 3DA, and is programmed by first writing a register address to port hex 3DA and then writing the data to port hex 3DA.

Any I/O 'write'-operations to hex address 3DA continuously toggle an internal address/data flip-flop. This internal flip-flop can be set to the address state by issuing an I/O 'read' instruction to port hex 3DA. An I/O 'read' instruction also 'reads' the status of the Video Gate Array. A description of each of the registers in the Video Gate Array follows.

Hex Address	Register
00	Mode Control 1
01	Palette Mask
02	Border Color
03	Mode Control 2
04	Reset
10-1F	Palette Registers

Video Gate Array Register Addresses

Mode Control 1 Register

This is a 5-bit 'write'-only register, it cannot be 'read'. Its address is 0 within the Video Gate Array. A description of this register's bit functions follows.

Bit 0	+HIBW/-LOBW
Bit 1	+Graphics/-Alpha
Bit 2	+B/W
Bit 3	+Video Enable
Bit 4	+16 Color Graphics

Mode Control 1 Register

Bit 0 This bit is 'high' (1) for all high-bandwidth modes. These modes are all modes which require the 64KB Memory and Display Expansion for a system total of 128K bytes of read/write memory. The high bandwidth modes are the 80 by 25 alphanumeric mode, the 640 by 200 4-color graphics mode, and the 320 by 200 16-color graphics mode. This bit is 'low' (0) for all low-bandwidth modes.

Bit 1 This bit is 'high' (1) for all graphics modes and is 'low' (0) for all alphanumeric modes.

Bit 2 When this bit is 'high' (1), the composite-video color-burst and chrominance are disabled, leaving only the composite intensity-levels for gray shades. When this bit is 'low' (0), the composite-video color is 'enabled'. This bit should be set 'high' for high-resolution black-and-white display applications.

Note: This bit has no effect on direct-drive colors.

Bit 3 When this bit is 'high' (1), the video signal is 'enabled'. The video signal should be 'disabled' when changing modes. When the video signal is 'disabled', the screen is forced to the border color.

Bit 4 This bit must be 'high' (1) for all 16-color graphics-modes. These modes are the 160 by 200 16-color graphics-mode and the 320 by 200 16-color graphics-mode.

Palette Mask Register

This is a 4-bit write-only register, it cannot be 'read'. Its address in the Video Gate Array is hex 01. A description of this register's bit functions follows.

Bit 0	–Palette Mask 0
Bit 1	–Palette Mask 1
Bit 2	–Palette Mask 2
Bit 3	–Palette Mask 3

Palette Mask Register

When bits 0-3 are 0, they force the appropriate palette address to be 0 regardless of the incoming color information. This can be used to make some information in memory a 'don't care' condition until it is requested.

In the 2-color and 4-color modes, the palette addresses should be 'masked' because only 1 or 2 color-lines contain valid information. For 4-color modes, the palette mask register should contain a hex 03 and, for 2-color modes, it should contain a hex 01.

Border Color Register

This is a 4-bit 'write'-only register, it cannot be 'read'. Its address in the Video Gate Array is hex 02. The following is a description of the register's bit functions:

Bit Number	Function
0	+ B (Blue) Border Color Select
1	+ G (Green) Border Color Select
2	+ R (Red) Border Color Select
3	+ I (Intensity) Border Color Select

Border Color Register

A combination of bits 0-3 selects the screen-border color as one of 16 colors, as listed in the "Summary of Available Colors" table in this section.

Mode Control 2 Register

This is a 4-bit, 'write'-only register, it cannot be 'read'. Its address inside the Video Gate Array is hex

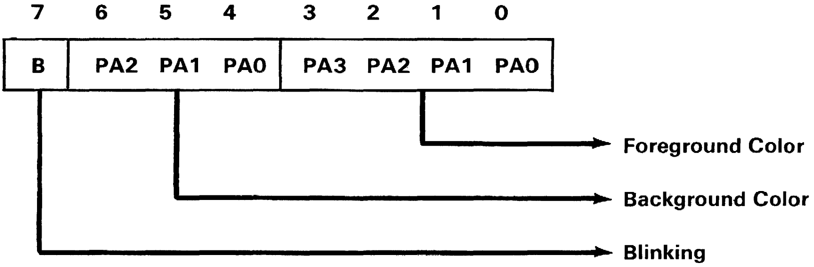
03. The following is a description of the register's bit functions:

Bit Number	Function
0 1 2 3	– Reserved = 0 + Enable Blink – Reserved = 0 + 2-Color Graphics

Mode Control 2 Register

Bit 0 This bit is reserved, but should always be programmed as a 0.

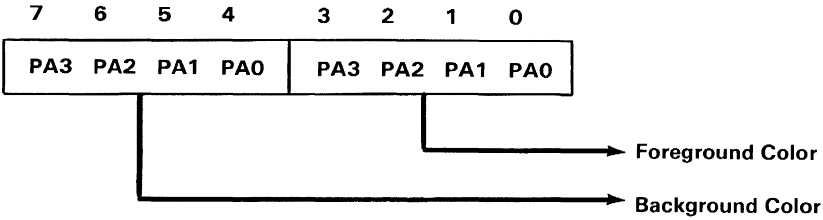
Bit 1 When this bit is 'high' (1) in the alphanumeric mode, the attribute byte has the following definition:



Where PA0 to PA3 are palette addresses.

Attribute Byte Definition (Part 1 of 2)

If the enable-blink bit is 'off' in the alphanumeric mode, the attribute byte takes on the following definition:



Attribute Byte Definition (Part 2 of 2)

If the enable-blink bit is on in a graphics mode, the high-order address of the palette (PA3) is replaced with the character-blink rate. This causes displayed colors to switch between two sets of colors.

If the colors in the lower half of the palette are the same as in the upper half of the palette, no color changes will occur. If the colors in the upper half of the palette are different from the lower half of the palette, the colors will alternately change between the 2 palette colors at the blink rate.

Only eight colors are available in the 16-color modes when using this feature. Bit 3 of the palette mask has no effect on this mode.

Bit 2 This bit is reserved, but should always be programmed as a 0.

Bit 3 This bit should be 'high' (1) when in the 640 by 200 2-color graphics-mode. It should be 'low' (0) for all other modes.

Reset Register

This is a 2-bit 'write'-only register, it cannot be 'read'. Its address inside the Video Gate Array is hex 04. The following is a description of the register's bit functions:

Bit 0	+Asynchronous Reset
Bit 1	+Synchronous Reset

Reset Register

Bit 0 When 'high' (1), this bit will issue an 'asynchronous reset' to the Video Gate Array. This will cause all memory cycles to stop and all output signals to be tri-stated. The 'asynchronous reset' should only be issued once at the system power-on time. This bit should be 'high' (1), the Video Gate Array and the 6845 programmed, and then it should be 'low' (0).

The system read/write memory (RAM) will not work until this power-on sequence is finished. After this power-on sequence, subsequent 'resets' should be 'synchronous resets'.

Note: Issuing an 'asynchronous reset' can cause the contents of RAM to be destroyed.

Bit 1 When 'high' (1), this bit will issue a 'synchronous reset' to the Video Gate Array. This will cause all memory cycles to stop and all output signals to stop. Bit 1 should be 'low' (0) before changing modes.

Before issuing a 'synchronous reset', the program should read 256 locations in RAM as every other location in 512 locations. The program should then issue the 'synchronous reset' and change the mode. This changes the Video Gate Array mode-control registers and the 6845 registers.

Next, the 'synchronous reset' should be removed and the 256 RAM locations should be 'read' again as above. This procedure will ensure system RAM data-integrity during mode changes. 'Synchronous resets' need only be issued when changing between high-bandwidth, and low- bandwidth modes. (Bit 0 in mode control 1 register)

Note: No accesses to RAM can be made while the video gate array is in a 'reset' state. 'Resets' must be done from code in ROM or EPROM's.

Palette Registers

There are sixteen 4-bit-wide palette-registers. These registers are 'write'-only, they cannot be 'read'. Their addresses in the Video Gate Array are from hex 10 to 1F.

Palette address hex 10 is accessed whenever the color code from memory is a hex 0, address hex 11 is accessed whenever the color code from memory is a hex 1, and so forth. A description of the color codes is in "Summary of Available Colors" in this section.

> **Note:** The palette address can be 'masked' by using the palette mask register.

The following is a description of the register's bit functions:

Bit Number	Function
0	+ Blue
1	+ Green
2	+ Red
3	+ Intensity

Palette Register Format

When loading the palette, the video is 'disabled' and the color viewed on the screen is the data contained in the register being addressed by the processor.

When the program has completed loading the palette, it must change the hex address to some address less than hex 10 for video to be 'enabled' again.

If a programmer does not wish a user to see the adverse effects of loading the palette, the palette should be loaded during the vertical-retrace time. The program must modify the palette and change the video gate array address to less than hex 10 within the vertical-retrace time. A vertical-retrace interrupt and a status bit are provided to facilitate this procedure.

Status Register

This is a 5-bit 'read'-only register, it cannot be 'written'. The internal address of the video gate array is a 'don't care' condition for the status-register read-operation. A description of the register's bit functions follows:

Bit 0	+Display Enable
Bit 1	+Light Pen Trigger Set
Bit 2	-Light Pen Switch Made
Bit 3	+Vertical Retrace
Bit 4	+Video Dots

Status Register

Bit 0 When 'high' (1), this bit indicates video is being displayed.

Bit 1 When 'high' (1), this bit indicates that a positive- going edge from the light pen input has set the light pen trigger. This trigger is 'low' (0) upon a system power-on, and may also be cleared by performing an I/O 'Out' command to address hex 3DB. No specific data is required, this action is address-activated.

Bit 2 This bit indicates the status of the light pen switch. The switch is not latched or debounced. When this bit is 'low' (0), the light pen switch is 'on'.

Bit 3 When 'high' (1), this bit indicates the vertical retrace is 'active'.

Bit 4 When 'high' (1), this bit indicates that video-dot information is available. The two low-order bits of the address register determine the video-dot information presented through the following logic:

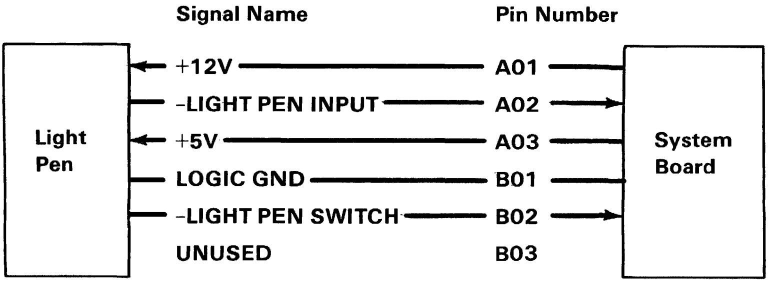
Address Register Bit 1	Address Register Bit 0	Video Dot Information Selected
0	0	Blue
0	1	Green
1	0	Red
1	1	Intensity

Address Register

This bit is provided for testing purposes. It verifies that video is occurring properly, and that the palette registers and all other 'write'-only registers are operating correctly.

Light Pen

A light pen can be used on the PC*jr* by connecting it to the six-pin connector for light pens on the back of the system board.



Connector Specifications

> **Note:** The light pen interface is set for RGBI (Red, Green, Blue, Intensity). Due to timing differences between different displays (Different phosphors take longer to turn on, and different circuits take longer to accomplish their task.) the row, column value returned from the CRT can vary. This difference must be compensated for through software.

Programming Considerations

Programming the 6845 CRT Controller

The 6845 has 19 accessible, internal registers, which are used to define and control a raster-scanned CRT display. One of these registers, the Index Register, is actually used as a pointer to the other 18 registers. It is a 'write'-only register, which is loaded from the processor by executing an 'Out' instruction to I/O address hex 3D4. The five least-significant-bits of the I/O bus are loaded into the Index Register.

In order to load any of the other 18 registers, the Index Register is first loaded with the necessary pointer; then the Data Register is loaded with the information to be placed in the selected register. The Data Register is loaded from the processor by executing an 'Out' instruction to I/O address hex 3D5.

The following table defines the values that must be loaded into the 6845-CRT-Controller registers to control the different modes of operation supported by the attachment:

Hex Addr.	Register #	Register Type	Units	I/O	Alphanumeric 40x25	Alphanumeric 80x25	Low/High Band Width Graphics
0	R0	Horizontal Total	Char.	Write Only	38	71	38/71
1	R1	Horizontal Display	Char.	Write Only	28	50	28/50
2	R2	Horizontal Sync Position	Char.	Write Only	2C	5A	2B/56
3	R3	Horizontal Sync Width	Char.	Write Only	06	0C	06/0C
4	R4	Vertical Total	Char. Row	Write Only	1F	1F	7F/3F
5	R5	Vertical Total Adjustment	Scan Line	Write Only	06	06	06/06
Note: All register values are given in hexadecimal.							

6845 Register Table (Part 1 of 3)

Hex Addr.	Register		Units	I/O	Alphanumeric		Low/High Band Width Graphics
	#	Type			40x25	80x25	
6	R6	Vertical Displayed	Char. Row	Write Only	19	19	64/32
7	R7	Vertical Sync Position	Char. Row	Write Only	1C	1C	70/38
8	R8	Interlace Mode	—	Write Only	02	02	02/02
9	R9	Maximum Scan Line Address	Scan Line	Write Only	07	07	01/03
A	R10	Cursor Start	Scan Line	Write Only	06	06	26/26
B	R11	Cursor End	Scan Line	Write Only	07	07	07/07
Note: All register values are given in hexademical.							

6845 Register Table (Part 2 of 3)

Hex Addr.	Register		Units	I/O	Alphanumeric		Low/High Band Width Graphics
	#	Type			40x25	80x25	
C	R12	Start Addr. (H)	—	Write Only	00	00	00/00
D	R13	Start Addr. (L)	—	Write Only	00	00	00/00
E	R14	Cursor Addr. (H)	—	Read/ Write	00	00	00/00
F	R15	Cursor Addr. (L)	—	Read/ Write	00	00	00/00
10	R16	Light Pen (H)	—	Read Only	NA	NA	NA/NA
11	R17	Light Pen (L)	—	Read Only	NA	NA	NA/NA
Note: All register values are given in hexadecimal.							

6845 Register Table (Part 3 of 3)

CRT/Processor Page Register

This register is an 8-bit 'write'-only register, that cannot be read. Its address is hex 3DF. The following is a description of the Register functions.

Bit Number	Description
0	CRT Page 0
1	CRT Page 1
2	CRT Page 2
3	Processor Page 1
4	Processor Page 2
5	Processor Page 3
6	Video Address Mode 0
7	Video Address Mode 1

CRT/Processor Page Register (Part 1 of 2)

CRT Page 0-2 These bits select which 16K byte memory-page between 00000 to hex 1FFFF is being displayed. If there is no expansion RAM in the system, the high- order bit is a 'don't care', and only 4 pages are supported. For graphics modes which require 32K bytes the low-order bit is a 'don't care'.

Processor Page 0-2 These bits select the 16K byte memory-page region where memory cycles to B8000 are redirected. If there is no expansion RAM installed in the system, the high-order bit is a 'don't care' and only 4 pages are supported.

Video Adr Mode 0-1 These bits control whether the row scan addresses are used as part of the memory address. These should be programmed as follows:

Video Address Mode		
1 (Bit 7)	**0 (Bit 6)**	**Resulting Modes**
0	0	All Alpha Modes
0	1	Low-Resolution-Graphics Modes
1	1	High-Resolution-Graphics Modes
1	0	Unused, Reserved

CRT/Processor Page Register (Part 2 of 2)

The following I/O devices are defined on the video color/graphics subsystem:

Hex Address	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Function of Register
3DA	1	1	1	1	0	1	1	0	1	0	Gate Array Address and Status Register
3DB	1	1	1	1	0	1	1	0	1	1	Clear Light Pen Latch
3DC	1	1	1	1	0	1	1	1	0	0	Preset Light Pen Latch
3D0,3D4	1	1	1	1	0	1	0	x	x	0	6845 Index Register
3D1,3D5	1	1	1	1	0	1	0	x	x	1	6845 Data Register
3DF	1	1	1	1	0	1	1	1	1	1	CRT, Processor Page Register
x = “don’t care” condition											

Video I/O Devices

Mode Selection Summary

Four registers of the Video Gate Array allow the user to access all the alphanumeric and graphics modes supported by the system ROM BIOS. The following table summarizes the modes and their register settings:

	Video Gate Array Reg.			
Mode	00	01	02	03
40 by 25 Alphanumeric Black-and-White	0C	0F	00	02
40 by 25 Alphanumeric Color	08	0F	00	02
80 by 25 Alphanumeric Black-and-White	0D	0F	00	02
80 by 25 Alphanumeric Color	09	0F	00	02
160 by 200 16-Color Graphics	1A	0F	00	00
320 by 200 4-Color Graphics	0A	03	00	00
320 by 200 4-Shade Black-and-White	0E	03	00	00
320 by 200 16-Color Graphics	1B	0F	00	00
640 by 200 2-Color Graphics	0E	01	00	08
640 by 200 4-Color Graphics	0B	03	00	00
Note: All values are given in hexadecimal.				

Mode Summary

Sequence of Events for Changing Modes

1. Determine the mode of operation.
2. Reset the 'video enable' bit in the Video Gate Array to disable video.
3. Program the 6845 CRT Controller to select the mode.
   Read 256 bytes of memory
   Reset gate array
4. Program the Video Gate Array registers.

Remove gate-array reset
Read 256 bytes of memory

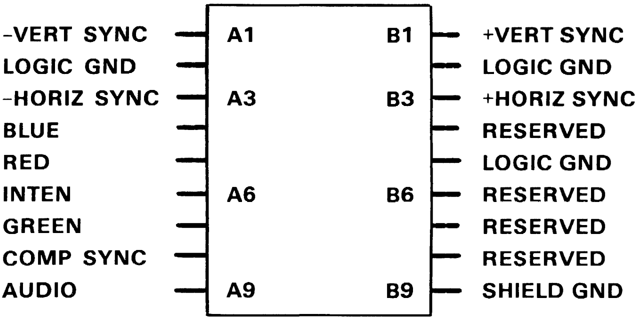
5. Re-enable video.

   Note: The gate array needs to be reset only when changing the high-bandwidth/low-bandwidth register.

Interrupt Information

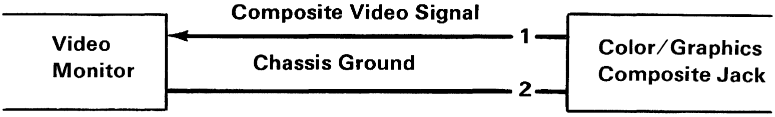
The Video Gate Array uses interrupt level 5 of the Intel 8259 to provide the vertical retrace interrupt to the system.

At Standard TTL Levels



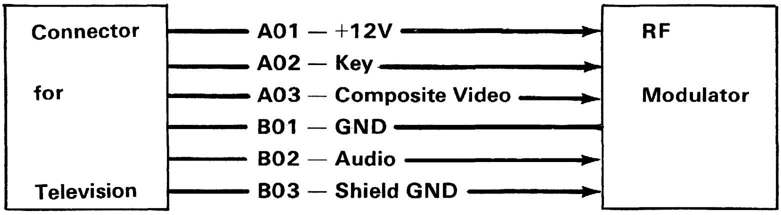
Connector Specifications

The direct-drive signals are standard TTL levels except the audio output which is a 1V peak-to-peak signal biased at 0V which can drive a 10K ohm or greater input-impedence.



Connector Specifications

The composite-video signal is 1V peak to peak biased at .7V with a 75 ohm load.



Television Connector Specifications

The Connector for Television connector has the composite-video signal at 1V peak to peak biased at .7V with a 75 ohm load. The connector also has the audio output which is 1V peak-to-peak signal biased at 0V which can drive a 10K ohm or greater input impedence.

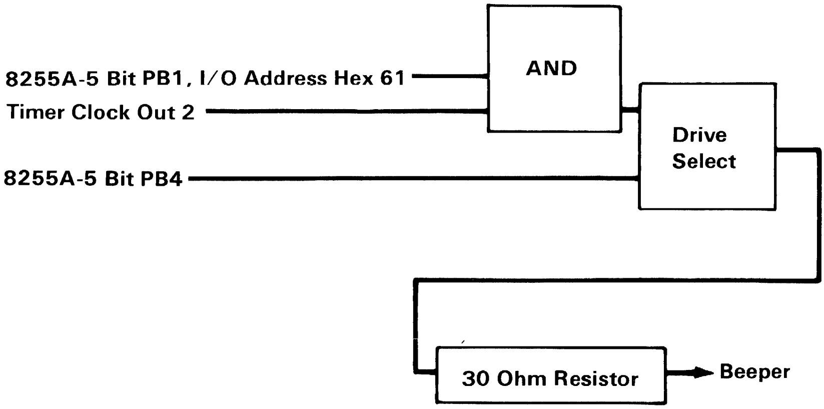
Notes:

Beeper

The system beeper is a small, piezoelectric- speaker, which can be driven from one or both of two sources. The two sources are:

- The 8255A-5 PPI output-bit PB1
- A timer clock out of an 8253-5 timer which has a 1.19 MHz-clock input. The timer gate is also controlled by an 8255-5 outport bit PBO.

  Note: The TI76496 Sound Generator cannot be directed through the beeper.

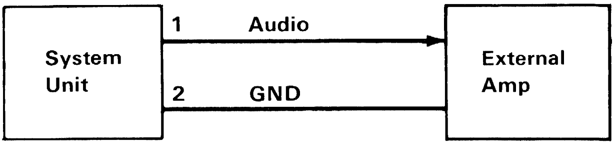


Beeper Block Diagram

Notes:

Sound Subsystem

The nucleus of the sound subsystem is an analog multiplexer (mpx) which allows 1 of 4 different sound sources to be selected, amplified, and sent to the audio outputs. The mpx and amplifier are configured so the amplifier's gain is unique to and consistent with each sound source. This provides a consistent level of output with any of the sound sources. The output of the amplifier is supplied to the IBM Connector for Television interface and external-amplifier interface. If an external speaker is used, an external amplifier must be used to drive it. The amplifier is configured as a single-pole low pass filter with a 3 dB cut-off frequency of 4.8 kHz. This filter is used to "round" off the corners of the square-wave signals. BIOS Power-on will initialize the sound subsystem to use the 8253 programmable-timer mode.



Connector Specifications

The audio output is a 1V peak-to-peak signal biased at 0V. It can drive a 10k ohm or greater input-impedence.

	Port Bits	
Source	PB6	PB5
Complex Sound Generator (TI 76496)	1	1
Programmable Timer (8253)	0	0
Cassette Audio	0	1
I/O Channel Audio	1	0

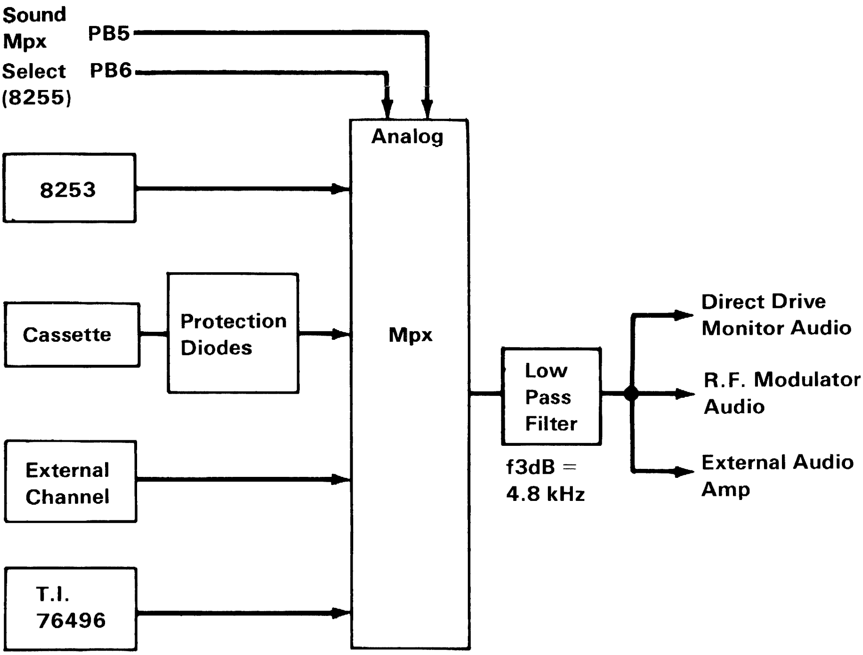
Port bits PB5 and PB6, of the 8255, control which source is selected.

Sound Sources

Complex Sound Generator

The Complex Sound Generator chip (SN76496N) has 3 programmable frequencies which may be mixed to form chords and a white noise generator which may also be mixed for special effects. Each of the 3 channels as well as the white noise generator can be independently attenuated. The processor controls the sound chip by writing to port hex C0.

The Sound Generator is described in greater detail later in this section. More information can be obtained by referring to Texas Instruments' data sheets and application notes.



Sound Block Diagram

Audio Tone Generator

Features

- 3 Programmable Tone-Generators
- Programmable White Noise
- Programmable Attenuation
- Simultaneous Sounds
- TTL Compatible
- 3.579 MHz Clock Input
- Audio Mixer

Processor to Sound-Generator Interface

The system microprocessor communicates with the SN76496N through the 8 data lines and 3 control lines

(WE, CE and READY). Each tone generator requires 10 bits of information to select the frequency and 4 bits of information to select the attenuation. A frequency update requires a double-byte transfer, while an attenuator update requires a single-byte transfer.

If no other control registers on the chip are accessed, a tone generator may be rapidly updated by initially sending both types of frequency and register data, followed by just the second byte of data for succeeding values. The register address is latched on the chip, so the data will continue going into the same register. This allows the 6 most-significant bits to be quickly modified for frequency sweeps.

Control Registers

The sound generator has 8 internal registers which are used to control the 3 tone generators and the noise source. During all data transfers to the sound generator, the first byte contains a 3-bit field which determines the destination control register. The register address codes are as follows:

Register Address Field			Destination Control Register
MSB R0	R1	LSB R2	
0	0	0	Tone 1 Frequency
0	0	1	Tone 1 Attenuation
0	1	0	Tone 2 Frequency
0	1	1	Tone 2 Attenuation
1	0	0	Tone 3 Frequency
1	0	1	Tone 3 Attenuation
1	1	0	Noise Control
1	1	1	Noise Attenuation

Register Address Field

	Reg. Addr.			Low Data			
1	R0	R1	R2	F6	F7	F8	F9
Bit 0 MSB			First Byte				Bit 7 LSB

	High Data						
0	X	F0	F1	F2	F3	F4	F5
Bit 0 MSB			Second Byte				Bit 7 LSB

Frequency (Double or Single Byte Transfer)

Frequency Generation

Each tone generator consists of a frequency-synthesis section and an attenuation section. The frequency-synthesis section requires 10 bits of information (hex F0-F9) to define half the period of the desired frequency (n). Hex F0 is the most-significant bit and hex F9 is the least-significant bit. This information is loaded into a 10-stage tone-counter, which is decremented at an N/16 rate where N is the input-clock frequency. When the tone counter decrements to 0, a borrow signal is produced. This borrow signal toggles the frequency flip-flop and also reloads the tone counter. Thus, the period of the desired frequency is twice the value of the period register.

The frequency can be calculated by the following:

$$f = \frac{N}{32n}$$

where N = ref clock in Hz (3.579 MHz)

n = 10-bit binary-number

Attenuator

	Reg. Addr.			Data			
1	R0	R1	R2	A0	A1	A2	A3
Bit 0 MSB				Second Byte			Bit 7 LSB

Update Attenuation (Single Byte Transfer)

The output of the frequency flip-flop feeds into a four-stage attenuator. The attenuator values, along with their bit position in the data word, are shown in the following figure. Multiple-attenuation control-bits may be 'true' simultaneously. Thus, the maximum theoretical attenuation is 28 dB typically.

Bit Position				
MSB A0	A1	A2	LSB A3	Weight
0	0	0	1	2dB
0	0	1	0	4dB
0	1	0	0	8dB
1	0	0	0	16db
1	1	1	1	OFF

Attenuator Values

Noise Generator

	Reg. Addr.					SHIFT	
	R0	R1	R2			NF0	NF1
1	1	1	0	X	FB		
MSB							LSB

Update Noise Source (Single Byte Transfer)

The noise generator consists of a noise source and an attenuator. The noise source is a shift register with an exclusive-OR feedback-network. The feedback network has provisions to protect the shift register from being locked in the zero state.

FB	Configuration
0	Periodic Noise
1	White Noise

Noise Feedback Control

Whenever the noise-control register is changed, the shift register is cleared. The shift register will shift at one of four rates as determined by the two NF bits. The fixed shift-rates are derived from the input clock.

Bits		
NF0	NF1	Shift Rate
0	0	N/512
0	1	N/1024
1	0	N/2048
1	1	Tone Generator #3 Output

Noise Generator Frequency Control

The output of the noise source is connected to a programmable attenuator.

Audio Mixer/Output Buffer

The mixer is a conventional operational-amplifier summing-circuit. It will sum the three tone-generator outputs, and the noise-generator output. The output buffer will generate up to 10 mA.

Data Transfer

The sound generator requires approximately 32 clock cycles to load the data into the register. The open collector READY output is used to synchronize the microprocessor to this transfer and is pulled to the false state (low voltage) immediately following the leading edge of CE. It is released to go to the true state (external pull-up) when the data transfer is completed.

This will insert approximately 42 wait states (8.9 μs) for each data transfer.

> **Warning:** Do not attempt to issue an I/O read operation to the TI76496 port (COH). Such an operation will cause the system to hang indefinitely.

> **Note:** If DMA is added to the system on the I/O channel, I/O WRITES to the 76496 will increase the latency time.

Notes:

Infra-Red Link

The infra-red link provides cordless communications between the keyboard and the system unit. Two infra-red-emitting diodes, mounted in the keyboard, transmit coded information to the system unit. The keyboard transmitter is fully discussed in "Cordless Keyboard" in this section. The infra-red receiver, which is located in the system unit, has an infra-red-sensitive device that demodulates the signal transmitted from the keyboard and sends it to the system.

Infra-Red Receiver

The receiver card measures 57.15 mm wide by 63 mm (2.25 in. by 2.50 in.) long. The infra-red receiver is mounted on the system board, component-side down, with two snap-in-type standoffs. Signal output and power input is through an 8-pin connector, located at the rear of the infra-red receiver. The infra-red-sensitive device is located on the front of the board and receives its input through an opening in the front of the system unit's cover. There is also an infra-red transmitter mounted on the receiver board for diagnostic purposes.

Functional Description

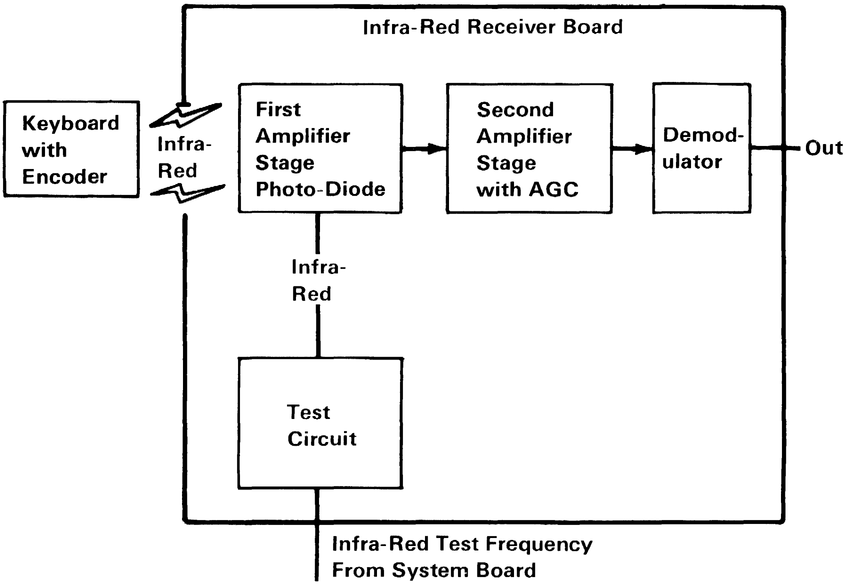
The following figure is the Infra-Red Receiver Block Diagram. During keyboard operation, the emitted light is modulated, transmitted, and received in the following sequence:

1. A key is pushed.

2. The data stream is sent using the infra-red-emitting diodes.
3. The receiver amplifies and processes the signal.
4. The demodulated signal is sent to the system board.

The signal received consists of an infra-red-light transmission modulated at 40 kHz.

An input is available (I/R Test Frequency) to the system for receiver-circuit-operational verification.


Infra-Red Receiver Board
Keyboard with Encoder
Infra-Red
First Amplifier Stage Photo-Diode
Second Amplifier Stage with AGC
Demodulator
Out
Infra-Red
Test Circuit
Infra-Red Test Frequency From System Board

Infra-Red Receiver Block Diagram

Application Notes

The Infra-Red Receiver Board can serve as a general-purpose infra-red-receiver, however, the demodulator timings are tailored to the needs of the system.

Programming Considerations

The serially-encoded word is software de-serialized by the 8088 processor on the system unit. The leading edge of the start bit will generate a non-maskable interrupt (NMI). Once the processor enters the NMI routine to handle the deserialization, the keyboard-data line is sampled and the processor waits to sample the trailing edge of the start bit. When the trailing edge of the start bit is sampled, the processor will wait for 310 μs and sample the first half of the first data bit. This delay causes the processor to sample in the nominal center of the first half of the first data bit. The processor then samples the keyboard data every half-bit cell-time. The sampling interval is 220 μs. The processor samples each half-bit-sample 5 times and will determine the logical level of the sample by majority rule. This enables the processor to discriminate against transient glitches and to filter out noise. The 8088 processor utilizes one 8255 PPI bit (PORT C BIT 6) and shares one 8253 timer channel (CHANNEL 1) to do the software de-serialization of the keyboard data. See the "Cordless Keyboard" in this section for more information on the data-transmission protocal.

Detectable Error Conditions

Errors	**Cause**
Phase Errors	The 1st half of the bit-cell sample is not equal to the inverse of the 2nd half of the bit-cell sample.
Parity Errors	The received encoded word did not maintain odd parity.

Note: Errors will be signaled by the processor with a short tone from the audio alarm or external speaker.

Operational Parameters

The operational distance from infra-red devices to the system should not exceed 6.1 meters (20 feet) (line-of-sight). Operational efficiency can be impaired by outside sources. These sources are, excessively-bright lights, and high-voltage lines, which include some TV sets. High-energy sources will generally cause an audible alarm within the system unit. These sources may downgrade the operational distance from the keyboard to the system. A keyboard cable is recommended if the above interference conditions are not controllable.

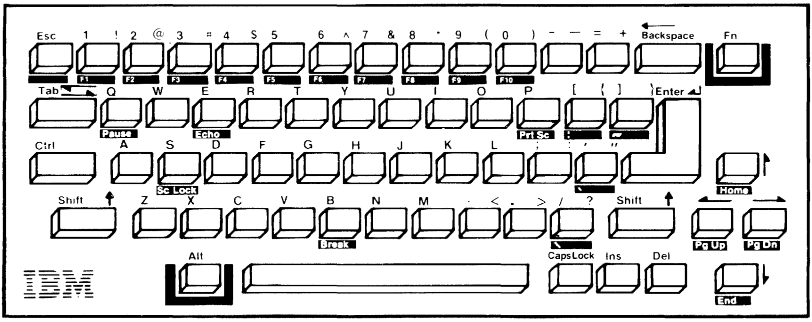
Pin	Signal	Input/Output
A01	+12 Volts	Input
A02	Ground	Input
A03	Ground-Shield	Input
A04	I.R. TEST FREQ.	Input
B01	GROUND	Input
B02	+5 Volts	Input
B03	-I.R. KBD DATA	Output
B04	GROUND	Input

Infra-Red Connector Specifications

IBM PC*jr* Cordless Keyboard

The keyboard is a low-profile, 62-key, detached keyboard with full-travel keys. The keys are arranged in a standard typewriter layout with the addition of a function key and cursor-control keys. The keybuttons are unmarked; however, an overlay is used to provide the keys' functional descriptions.

The following figure shows the layout of the cordless keyboard.



The keyboard is battery powered and communicates to the system unit with an infra-red (IR) link. The infra-red link makes the remote keyboard a truly portable hand-held device. An optional-cord connection to the system unit is available. Power is sent to the keyboard and serially-encoded data received by the system unit through the optional cord. When connected, the cord's keyboard-connector removes the battery power and the -CABLE CONNECT signal disables the infra-red-receiver circuit. The disabling of the circuit also allows other infrared devices to be used without interfering with the system. The data which is received through the IR link or by the cord, have the same format.

The keyboard interface is designed to maximize system-software flexibility in defining keyboard operations such as shift states of keys, and typematic operation. This is accomplished by having the keyboard return scan codes rather than American National Standard Code for Information Interchange (ASCII) codes. The scan codes are compatible with Personal Computer and Personal Computer XT scan codes at the BIOS interface level. All of the keys are typematic and generate both a make and a break scan-code. For example, key 1 produces scan code hex 01 on make and code hex 81 on break. Break codes are formed by adding hex 80 to the make codes. The keyboard I/O driver can define keyboard keys as shift keys or typematic, as required by the application.

The microprocessor in the keyboard performs keyboard scanning, phantom-key detection, key debounce, buffering of up to 16 key-scan-codes, and transfer of serially-encoded data to the system unit. The keyboard microprocessor is normally in a standby power-down mode until a key is pressed. This causes the microprocessor to scan the keyboard. The microprocessor then transmits the scan code, and re-enters the power-down mode if its buffer is empty and no keys are pressed.

The keyboard electronics is designed with low-power CMOS integrated-circuitry for battery power operation. Four AA-size batteries are required. Because the keyboard is normally in the standby power-down mode, which uses very little power, no on/off switch is needed.

Unlike other keyboards in the IBM Personal Computer family, the IBM PC*jr* Cordless Keyboard has phantom-key detection. Phantom-key detection occurs when invalid combinations of three or more keys are pressed simultaneously, causing a hex 55 scan-code to be sent to the keyboard's processor. The phantom-key scan-code instructs the keyboard's processor to ignore all of the keys that were pressed at that time. BIOS ignores the resulting scan-code that is sent to it.

The keyboard-cord connector provides a battery-disconnect function and also disables the infra-red-transmission circuitry when the mating plug for the modular jack is connected.

Note: See "Keyboard Encoding and Usage" in Section 5, for scan codes and further information.

Transmitter

Serially encoded words are transmitted to the system unit using the Infra-Red Link or the cable link. Encoded words are sent to the system unit with odd parity. Both the Infra-Red Link and the cable link use biphase serial-encoding and each is a simplex link.

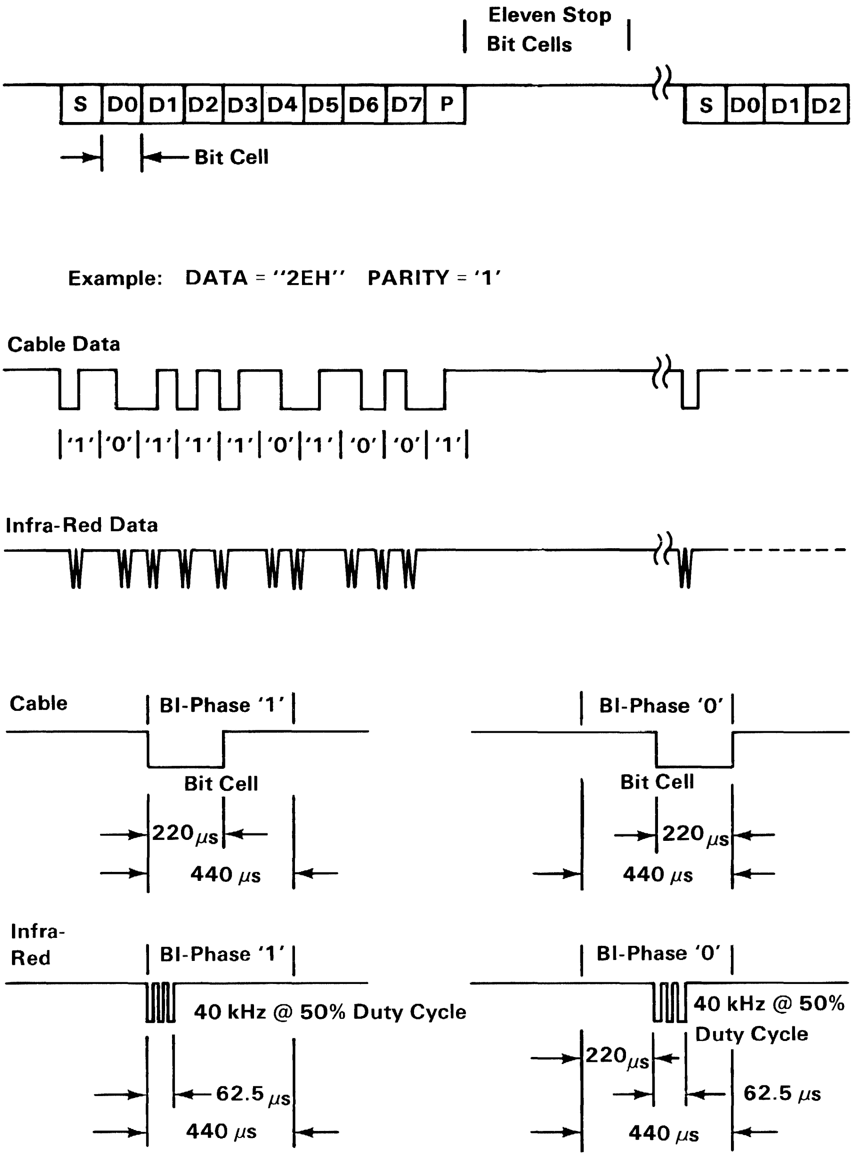
The 80C48 microprocessor does the biphase serial encoding with a bit cell of 440 μs. A biphase logically-encoded 1 is transmitted as logical 1 for the first half of the bit cell time and as a logical 0 for the second half of the bit cell. A biphase logically-encoded 0 is transmitted as a logical 0 for the first half of the bit cell time and as a logical 1 for the second half of the bit cell.

Each logical 1 transmission for the Infra-Red Link consists of a 40 kHz carrier burst at a 50% duty cycle.

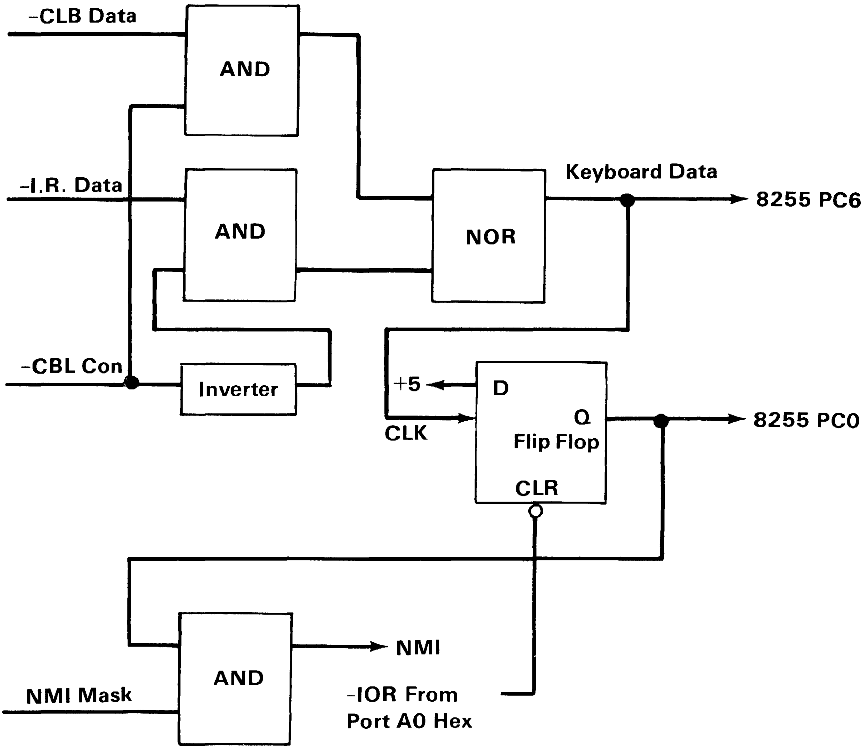
First Bit	Start Bit
Second Bit	Data Bit 0 (Least Significant Bit)
Third Bit	Data Bit 1
Fourth Bit	Data Bit 2
Fifth Bit	Data Bit 3
Sixth Bit	Data Bit 4
Seventh Bit	Data Bit 5
Eight Bit	Data Bit 6
Ninth Bit	Data Bit 7 (Most Significant Bit)
Tenth Bit	Parity Bit
Eleventh Bit	Stop Bit

Data Stream Sequence

Eleven stop bits are inserted after every scan-code transmission. This is to allow some processor bandwidth between keystrokes to honor other types of interrupts, such as serial and time-of-day.



Keyboard Transmission Timing



Keyboard Interface Logic

Program Cartridge and Interface

The Program Cartridge allows the addition of ROM to the system without removing the cover by plugging it , into either of two slots in the front of the machine.

The 48 by 72 mm (2 by 3 inch) cartridge can hold one or two 32K byte by 8 ROMS (64K bytes total) of program storage. Smaller ROMS such as the 8K byte by 8 modules can be used in the cartridge. When a smaller module is used, the higher address lines are not used. To allow two smaller modules to be mapped to adjacent memory segments, each module's contents is addressed to multiple adjacent-memory segments, within the addressable range of the module's socket (32k).

Program Cartridge Slots

The Program Cartridge is designed to plug into either of two identical slots in the front of the machine. Each slot has 15 address signals, 8 data signals, 6 chip selects, 2 control signals, and power. Cartridge selection is accomplished by the chip selects, each of which addresses one of the high 32K memory-blocks. Each cartridge uses up to two of the six chip selects. Selection is determined on the basis of the intended use of the cartridge. This is done at the factory.

Two of the chip selects are used by the internal system-ROM. These two signals can be used to allow the internal ROM to be replaced by a Program Cartridge. This allows the machine to assume a different personality from the standard machine. To use this option of mapping the internal-ROM space to a cartridge, the Base-ROM-in-Cartridge function must be inserted. This function is a factory-installed signal-jumper manufactured into particular program-cartridges that are intended to replace the system ROM.

> **Note:** When the cartridge is inserted or removed with the system turned on, the system will 'reset' and go through a warm power-up. Any data in the system RAM will be lost.

Cartridge Storage Allocations

A. The following conventions will be followed for "Initial Program Loadable" program cartridges:

Location	Contents
0	055H
1	0AAH
2	Length
3,4,5	Jump to Initialize Code
6	0
Last 2 Addresses	CRC Bytes

Storage Conventions

- Locations 0 and 1 contain the word hex 55AA. This is used as a test for the presence of the cartridge during the configuration- determination portion of the power-on routines.
- Location 2 contains a length indicator representing the entire address space taken by the ROM on the cartridge. The algorithm for determining the

contents of this byte is (length/512). The contents of this byte is used by the CRC (cyclic-redundancy-check) routine to determine how much ROM to check.

- Location 3 contains the beginning of an initialization routine that is reached by a 'Long' call during the power-on sequence. For cartridges that are 'IPL-able' (BASIC or assembler program) this routine should set the INT hex 18 vector to point to their entry points. Other types of cartridges (BASIC or whatever) should merely 'return' to the caller. Setting the INT hex 18 vector will enable transfer of control to the cartridge program by the IPL routine.
- This location 6 should be 00.
- CRC bytes: The last two locations of the address space used by the cartridge must be blank. CRC characters will be placed in these bytes when the cartridge is built. See the routine at label "CRC Check", in the BIOS listing for the CRC algorithm.

B. The following conventions will be followed for cartridges that wish to be recognized by DOS 2.1 as containing code associated with DOS command words:

Location	Contents
0	055H
1	0AAH
2	Length
3-5	Jump to Initialize
6	Command Name Length (Offset Y-Offset Z)
Z	First Character in Command Name
Y	Last Character in Command Name
W	Word Pointing to Routine that is Jumped to if "Name" is Typed
X	Next Command Name Length or "00" if No More Command Names
Last 2 Addresses	CRC Bytes

DOS Conventions

- Locations 0 and 1 contain the word hex 55AA. This is used as a test for the presence of the cartridge during the configuration- determination portion of the power-on routines.
- Location 2 contains a length indicator representing the entire address space taken by the ROM on the cartridge. The algorithm for determining the contents of this byte is (length/512). The contents of this byte is used by the CRC routine to determine how much ROM to check.
- Location 3 contains a 'jump' to the initialization code for this ROM. (May just be a 'Far Return')
- Starting at location 6 may be a sequence of command name pointers consisting of 1: Count of length name, 2: Name in ASCII, and 3: Word

containing offset within this segment to the code that is entered when this name is called. There can be as many names as desired, providing that a hex 00 is placed in the count field following the last name pointer. If a cartridge has a routine called 'TEST' at location hex 0FB5 (offset from start of segment that the cartridge is in) that needs to be executed when 'test' is entered as a DOS command the entry at location 6 would be hex 04,54,45,53,54,B5,0F.

- CRC bytes: The last two locations of the address space used by the cartridge must be blank. CRC characters will be placed in these bytes when the cartridge is built. See the routine at label "CRC Check", in the BIOS listing for the CRC algorithm.

C. The following conventions will be followed for cartridges that wish to be recognized by "Cartridge BASIC" as containing interpretable-BASIC Code:

- The cartridge-chip selects must address hex D0000 since the BASIC cartridge addresses hex E0000. When "Cartridge BASIC" is activated, it will check for a second cartridge program at hex D0000. If the second cartridge is present and formatted properly, then the BASIC code is loaded into RAM and run.
- The format for this interpretable-BASIC code must be as follows:

Location	Contents
0	055H
1	0AAH
2	Length
3	0CBH
4	0AAH
5	055H
6	0
7	0FFH if unprotected Basic program or 0FEH if protected Basic program
8	Start of interpretable Basic code
n	0FFH Padding to next 2048 byte boundary
Last 2 Addresses	CRC Bytes

Cartridge Format

1. Locations 0 and 1 contain the word hex 55AA. This is used as a test for the presence of the cartridge during the configuration-determination portion of the power-on routines.
2. Location 2 contains a length indicator representing the entire address space taken by the ROM on the cartridge. The algorithm for determining the contents of this byte is (length/512). The contents of this byte is used by the CRC routine to determine how much ROM to check.
3. Location 3 must be hex 0CB for a 'far return' instruction.

4. Locations 4 and 5 contain the word hex AA55. This is used as a test for the presence of the second cartridge by "Cartridge Basic".
5. Location 6 must be a 0 to follow the DOS conventions.
6. Location 7 can be either hex FF to indicate an unprotected BASIC program, or hex FE to indicate a protected program.
7. Location 8 must be the start of the BASIC program. It must be interpretable Basic and not compiled. Also, at the end of the program PAD to the next 2048 byte boundary with hex 0FF.
8. CRC bytes: The last two locations of the address space used by the cartridge must be blank. CRC characters will be placed in these bytes when the cartridge is built. See the routine at label "CRC Check", in the BIOS listing for the CRC algorithm.

ROM Module

The ROM modules used are 250 ns devices. Typical modules are the Mostek MK37000 and MK38000, the TMM 23256, the SY23128, and other compatible devices.

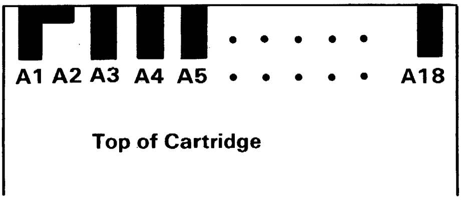
ROM Chip Select	Hex Address Space	Typical Use
CS0	X	Not Used
CS1	X	Not Used
CS2	D0000-D7FFF	Optional Cartridge ROM #2
CS3	D8000-DFFFF	Optional Cartridge ROM #1
CS4	E0000-E7FFF	Standard Cartridge ROM #2
CS5	E8000-EFFFF	Standard Cartridge ROM #1
CS6	F0000-F7FFF	System Board ROM #2
CS7	F8000-FFFFF	System Board ROM #1

ROM Chip Select Table

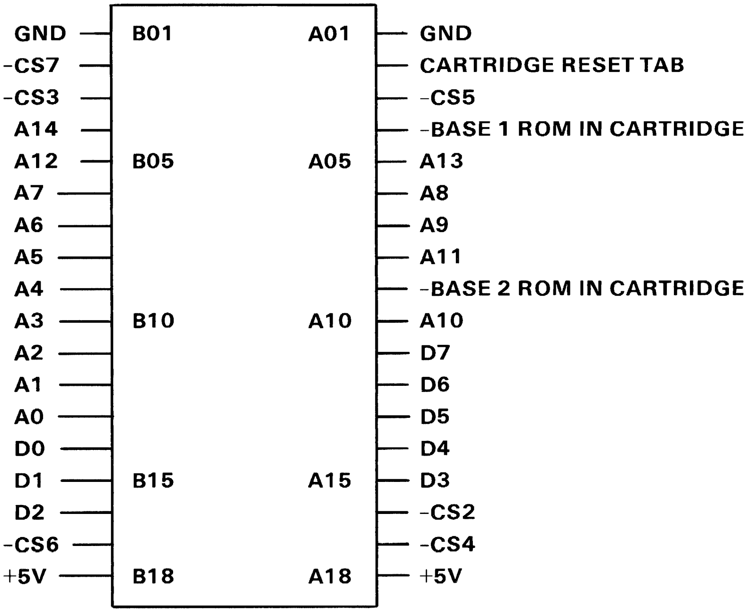
Signal	I/O	Description
A0 - A14	0	Processor Address lines A0 - A14
D0 - D7	I	Processor Data lines

-CS2 THRU -CS7	0	These chip-select lines are used to select ROM modules at different addresses. The addresses for each chip-select are shown in the ROM-chip select-table. -CS6 and -CS7 are used on the system board for BIOS, Power-On-Self-Test (POST) and cassette-basic ROMs. In order to use these chip selects on a cartridge, -BASE 1 ROM IN CARTRIDGE or -BASE 2 ROM IN CARTRIDGE must be pulled 'low'
-BASE 1 ROM IN CARTRIDGE	I	This line when pulled 'low' instructs the system board to de-gate the ROM module from hex F8000 - FFFFF on the system board. This ROM module can then be replaced by a ROM module on the cartridge by using -CS7.
-BASE 2 ROM IN CARTRIDGE	I	This line when pulled 'low' instructs the system board to de-gate the ROM module from hex F0000 - F7FFF on the system board. This ROM module can then be replaced by a ROM module on the cartridge by using -CS6.

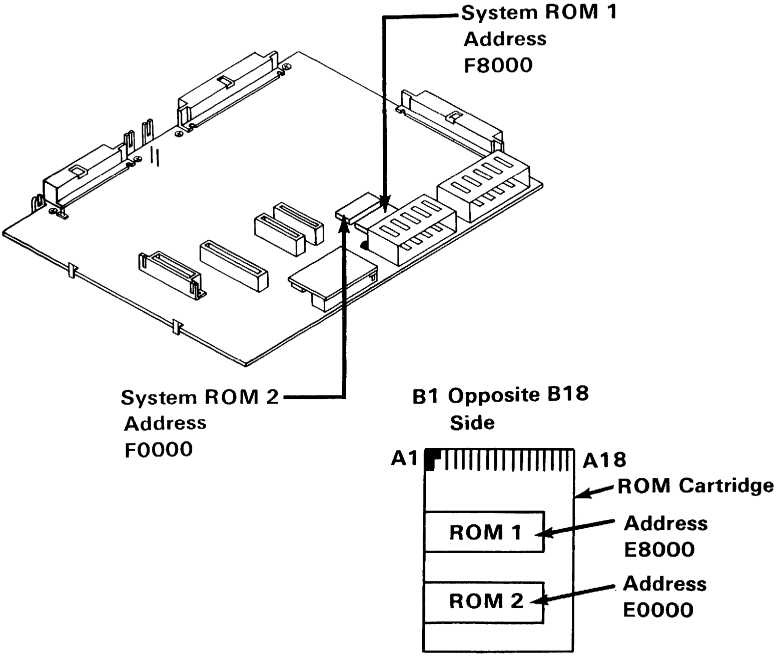
Cartridge Reset Tab	I	This input when 'low' causes a 'reset' to the system. The system will remain 'reset' until this line is brought back 'high'. This tab is usually wired with an L shaped land pattern to the GND at A02 which provides a momentary 'reset' when a cartridge is inserted or removed.



Momentary Reset Land



Connector Specification



Cartridge ROM Locations

Games Interface

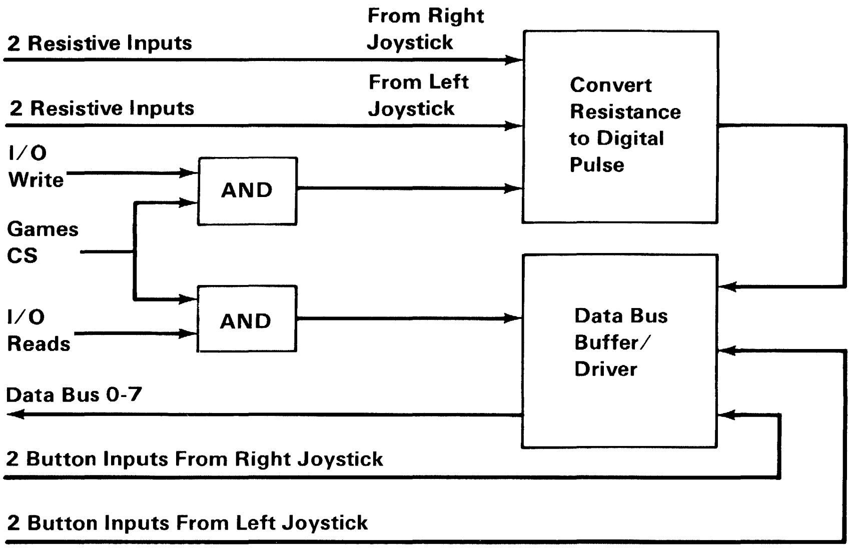
Interface Description

The Game Interface has two connectors located at the rear of the System unit for four paddles (two per connector) or two joysticks. Each connector has four input lines: two digital inputs and two resistive inputs. All the inputs are 'read' with one 'IN' from address hex 201. The interface, plus system software, converts the present resistive value to a relative paddle or joystick-position. On receipt of an output signal, four timing circuits are started. By determining the time required for the circuit to time out (a function of the resistance), the paddle or joystick position can be determined.

The four digital inputs each have a 1K ohm resistor to pull the voltage up to +5V. With no drive on these inputs, a 1 is read. For a 0 reading, the inputs must be pulled to ground.

The four resistive inputs are converted to a digital pulse with a duration proportional to the resistive load, according to the following equation:

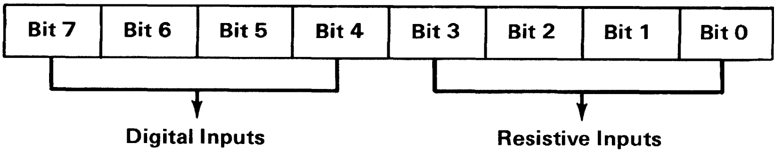
```
Time = 24.2 µs + 0.011 (r) µs
Where r is the resistance in ohms
```



Games Interface Block Diagram

Any program application must first begin the conversion by an 'OUT' to address hex 201. An 'IN' from address hex 201 will show the digital pulse go 'high' and remain 'high' for the duration according to the resistance value. All four bits (Bit 3 through Bit 0) function in the same manner. Each bits digital pulse goes high simultaneously and resets independently according to the input resistance value.

Input from Address Hex 201



Input From Address Hex 201

Joysticks typically have one or two buttons and two variable resistances each. The variable resistances are mechanically linked to have a range from 0 to 100k ohms. One variable resistance indicates the X coordinate and the other variable resistance indicates the Y coordinate. The joysticks are attached to give the following input data:

Joystick B		Joystick A		Joystick B		Joystick A	
Button #2	Button #1	Button #2	Button #1	Coord. Y	Coord. X	Coord. Y	Coord. X
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Joystick Input Data

The game paddles have one button each and one variable resistance each. The variable resistance is mechanically linked to have a range from 0 to 100k ohms. The paddles are attached to give the following input data.

Buttons				Coordinates			
Paddle D	Paddle C	Paddle B	Paddle A	Paddle D	Paddle C	Paddle B	Paddle A
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Paddle Input Data

Pushbuttons

The pushbutton inputs are 'read' by an 'IN' from address hex 201. These values are seen on data bits 7 through 4. These buttons default to an 'open' state and are 'read' as 1. When a button is pressed, it is 'read' as 0.

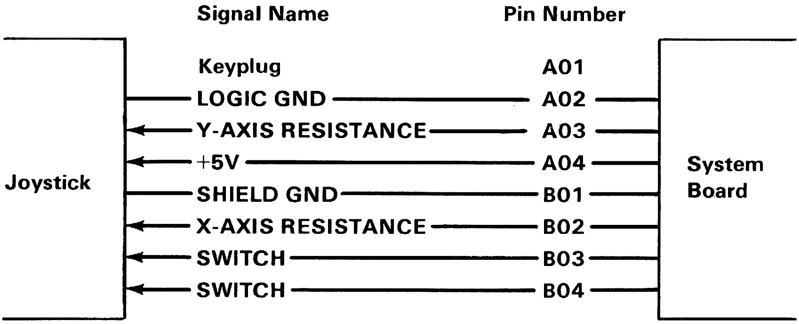
> **Note:** Software should be aware that these buttons are not debounced in hardware.

Joystick Positions

The joystick position is indicated by a potentiometer for each coordinate. Each potentiometer has a range from 0 to 100k ohms that varies the time constant for each of the four one-shots. As this time constant is set at different values, the output of the one-shot will be of varying durations.

All four one-shots are fired simultaneously by an 'OUT' to address hex 201. All four one-shot outputs will go 'true' after the fire pulse and will remain 'high' for varying times depending on where each potentiometer is set.

These four one-shot outputs are 'read' by an 'IN' from address hex 201 and are seen on data bits 3 through 0.



Connector Specification

Notes:

Serial Port (RS232)

The PC*jr* serial port is fully programmable and supports asynchronous communications only. It will add and remove start bits, stop bits, and parity bits. A programmable baud-rate generator allows operation from 50 baud to 4800 baud. Five, six, seven or eight bit characters with 1, 1-1/2, or 2 stop bits are supported. A fully-prioritized interrupt-system controls transmit, receive, line status and data-set interrupts. Diagnostic capabilities provide loopback functions of transmit/receive and input/output signals.

The nucleus of the adapter is a 8250A LSI chip or functional equivalent. Features in addition to those previously listed are:

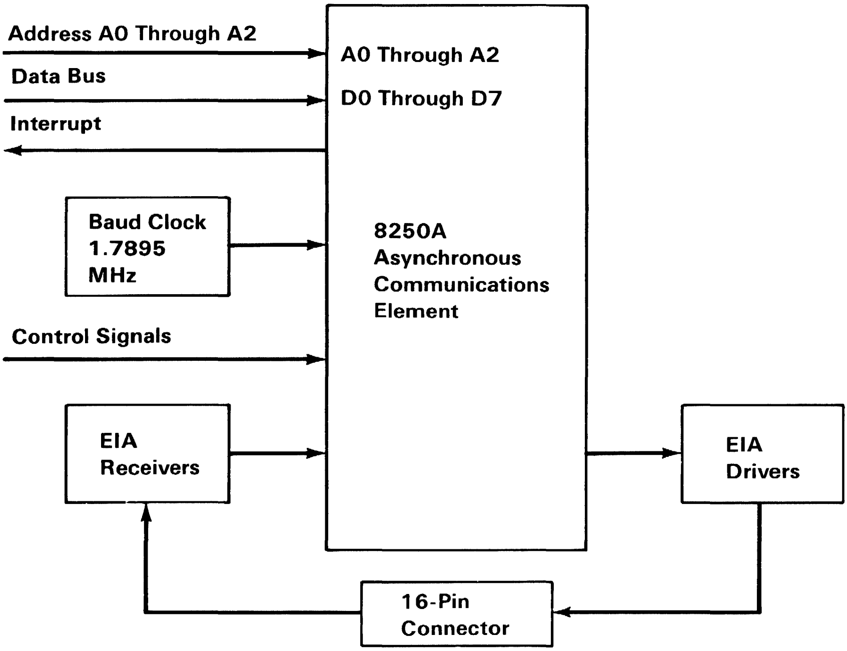
- Full double-buffering eliminates the need for precise synchronization
- Independent receiver clock input
- Modem control functions: clear to send (CTS), request to send (RTS), data set ready (DSR), data terminal ready (DTR)
- Even, odd, or no-parity-bit generation and detection
- False start bit detection
- Complete status reporting capabilities
- Line-break generation and detection
- Break, parity, overrun, and framing error simulation
- Full prioritized interrupt system controls

All communications protocol is a function of the system ROM and must be loaded before the adapter is operational. All pacing of the interface and control-signal status must be handled by the system software. It should be noted that Asynchronous (Async) receive operations cannot overlap diskette operation since all but the Diskette Interrupt are masked 'off' during diskette operations. If Async receive operations are going to be overlapped with keyboard receive operations, the Async Receiver rate cannot exceed 1200 baud. This is due to the processor deserialization of the keyboard. See IBM PC*jr* Cordless Keyboard in this section for more information.

Programming Note: Due to the read/write cycle-time of the 8250A, it is recommended that back-to-back I/O operations to the 8250A be avoided. A good Programming Technique would be to insert a short 'jump' between every consecutive 8250 I/O instruction. This action will flush the queue and provide 15 clock periods between I/O operations.

> **Note:** This note only applies to programmers using the 8250A directly. It is STRONGLY suggested that the user not communicate directly with the physical hardware, but use the system BIOS instead.
>
> **Note:** It is important to note that when the IBM PC*jr* has the Internal Modem installed it is logically COM1 and the RS232 serial port is logically COM2 in BIOS, DOS, and BASIC. Without the Internal Modem installed the RS232 serial port is logically addressed as COM1 in BIOS, DOS ,and BASIC even though its address is still hex 2F8 using Interrupt level 3.

The following figure is a Serial Port Block Diagram:



Serial Port Block Diagram

Modes of Operation

The different modes of operation are selected by programming the 8250A asynchronous communications element. This is done by selecting the I/O address (hex 2F8 to 2FF) and 'writing' data out to the card. Address bits A0, A1, and A2 select the different registers that define the modes of operation. Also, the divisor-latch access-bit (bit 7) of the line-control register is used to select certain registers.

I/O Decode (in Hex)	Register Selected	DLAB State
2F8	TX Buffer	DLAB=0 (Write)
2F8	RX Buffer	DLAB=0 (Read)
2F8	Divisor Latch LSB	DLAB=1
2F9	Divisor Latch MSB	DLAB=1
2F9	Interrupt Enable Register	DLAB=0
2FA	Interrupt Identification Registers	(Don't Care)
2FB	Line Control Register	(Don't Care)
2FC	Modem Control Register	(Don't Care)
2FD	Line Status Register	(Don't Care)
2FE	Modem Status Register	(Don't Care)
2FF	Scratch Register	(Don't Care)

I/O Decodes

Address Range hex 2F8 - 2FF

Note: The state of the divisor-latch access-bit (DLAB), which is the most-significant bit of the line-control register, affects the selection of certain 8250A registers. The DLAB must be set 'high' by the system software to access the baud-rate-generator divisor latches.

Interrupts

One interrupt line is provided to the system. This interrupt is IRQ3 and is 'positive active'. To allow the serial port to send interrupts to the system, bit 3 of the modem control register must be set to 1 'high'. At this point, any of the following interrupt types 'enabled' by bits in the interrupt-enable register will cause an interrupt: Receiver-line status, Received Data available, Transmitter-Holding-Register empty, or Modem Status.

Interface Description

The communications adapter provides an EIA RS-232C electrically-compatible interface. One 2 by 8-pin Berg connector is provided to attach to various peripheral devices.

The voltage interface is a serial interface. It supports data and control signals as follows:

Pin A04	Transmit Data
Pin A08	Receive Data
Pin A03	Request to Send
Pin A07	Clear to Send
Pin A06	Data Set Ready
Pin B02-B08	Signal Ground
Pin A05	Carrier Detect
Pin A02	Data Terminal Ready
Pin B01	Shield Ground

The adapter converts these signals to/from TTL levels to EIA voltage levels. These signals are sampled or generated by the communications-control chip. These signals can then be sensed by the system software to determine the state of the interface or peripheral device.

> **Note:** The above nomenclature describes the communications adapter as a DTE (Data Terminal Equipment) device. Suitable adapters must be used to attach other devices such as serial printers.

> **Note:** Ring Indicate is not supported on the PC*jr*.

Voltage Interchange Information

Interchange Voltage	Binary State	Signal Condition	Interface Control Function
Positive Voltage =	Binary (0)	= Spacing	= On
Negative Voltage =	Binary (1)	= Marking	= Off

Voltage Interchange Information

```
                          Invalid Levels
  +15 Vdc       ------------------------------
                           On Function
   +3 Vdc       ------------------------------

    0 Vdc                 Invalid Levels

   -3 Vdc       ------------------------------
                          Off Function
  -15 Vdc       ------------------------------
                          Invalid Levels
```

The signal will be considered in the 'marking' condition when the voltage on the interchange circuit, measured at the interface point, is more negative than

-3 Vdc with respect to signal ground. The signal will be considered in the 'spacing' condition when the voltage is more positive than +3 Vdc with respect to signal ground. The region between +3 Vdc and -3 Vdc is defined as the transition region, and considered an invalid level. The voltage which is more negative than -15 Vdc or more positive than +15 Vdc will also be considered an invalid level.

During the transmission of data, the 'marking' condition will be used to denote the binary state 1, and the 'spacing' condition will be used to denote the binary state 0.

For interface control circuits, the function is 'on' when the voltage is more positive than +3 Vdc with respect to signal ground and is 'off' when the voltage is more negative than -3 Vdc with respect to signal ground.

For detailed information regarding the INS8250A Communications Controller, refer to "Bibliography".

Output Signals

Output 1 (OUT 1), Pin 34: Output 1 of the 8250A is not supported in PC*jr* hardware.

Output 2 (OUT 2), Pin 31: Output 2 of the 8250A is not supported in PC*jr* hardware.

Accessible Registers

The INS8250A has a number of accessible registers. The system programmer may access or control any of the INS8250A registers through the processor. These registers are used to control INS8250A operations and to transmit and receive data. For further information regarding accessible registers, refer to "Bibliography".

INS8250A Programmable Baud Rate Generator

The INS8250A contains a programmable baud rate generator that is capable of taking the clock input (1.7895 MHz) and dividing it by any divisor from 1 to (65535). The output frequency of the Baud Rate Generator is 16 x the baud rate [divisor number = (frequency input) / (baud rate x 16)]. Two 8-bit latches store the divisor in a 16-bit binary- format. These divisor latches must be loaded during initialization in order to ensure desired operation of the baud rate generator. Upon loading either of the divisor latches, a 16-bit baud-counter is immediately loaded. This prevents long counts on initial load.

The following figure illustrates the use of the baud rate generator with a frequency of 1.7895 MHz. For baud rates of 4800 and below, the error obtained is minimal.

> **Note:** The maximum operating frequency of the baud generator is 3.1 MHz. In no case should the data rate be greater than 4800 baud.

Desired Baud Rate	Divisor Used to Generate 16x Clock (Decimal)	(Hex)	Percent Error Per Bit Difference Between Desired and Actual
50	2237	8BD	.006
75	1491	5D3	.017
110	1017	1A1	.023
134.5	832	167	.054
150	746	12C	.050
300	373	175	.050
600	186	BA	.218
1200	93	5D	.218
1800	62	3E	.218
2000	56	38	.140
2400	47	2F	.855
3600	31	1F	.218
4800	23	17	1.291

Baud Rate at 1.7895 MHz

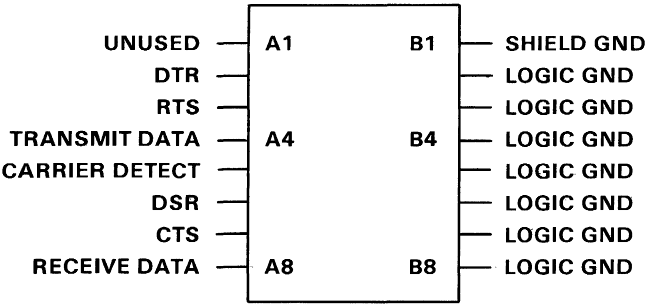
Note: These divisions are different than that used in the IBM Personal Computer. For portability, all initialization should be done through the system BIOS.

Note: Receive rates should not exceed 1200 baud if the receive operation is overlapped with keyboard keystrokes.

The following Assembly language sample program initializes the 8250. The baud rate is set to 1200 baud. It's data word is defined: 8 bits long with 1 stop bit odd parity.

```
BEGIN   PROC    NEAR
        MOV     AL,80H     ;  SET DLAB = 1
        MOV     DX,2FBH    ;  To Line Control Register
        OUT     DX,AL
        JMP     $+2        ;  I/O DELAY
        MOV     DX,2F8H    ;  Point to LSB of Divisor Latch
        MOV     AL,5DH     ;  This is LSB of Divisor
        OUT     DX,AL
        JMP     $+2        ;  I/O DELAY
        MOV     DX,2F9H    ;  Point to MSB of Divisor Latch
        MOV     AL,0       ;  This is MSB of Divisor
        OUT     DX,AL
        JMP     $+2        ;  I/O DELAY
        MOV     DX,2FBH    ;  Line Control Register
        MOV     AL,0BH     ;  8 Bits/Word, 1 Stop Bit,
                              Odd Parity, DLAB = 0
        OUT     DX,AL
        JMP     $+2        ;  I/O DELAY
        MOV     DX,2F8H
        IN      AL,DX      ;  In Case Writing to Port LCR Caused
                           ;  Data Ready to go high
        ENDP
BEGIN
```

Assembly Language Sample Program



Connector Specifications

System Power Supply

The system power supply is a 33 Watt, three voltage-level, two-stage supply. The first stage is an external power transformer that provides a single-fuse protected, extra low, ac-voltage output. The power cord is 3.08 meters (10.16 feet) long. The second stage is an internal, printed-circuit board, which is vertically mounted into the system board. The second stage converts the transformer's ac-output into three dc-output levels.

The amount of power available on the I/O connector for a machine that is fully configured with internal features is 400 mA of +5 Vdc, 0 mA of +12 Vdc and 0 mA of -6 Vdc.

Power is supplied to the system board through a printed-circuit-board edge-connector. The diskette drive is powered through a separate four-pin connector mounted on the front edge of the Power Board. The power for the diskette drive fan is provided by a three-pin Berg-type connector mounted directly below the diskette-drive connector. Power is removed from the system board and diskette drive by a switch mounted on the rear of the Power Board. Both the switch and the transformer connector are accessible from the rear of the system.

Operating Characteristics

Power Supply Input Requirements

Voltage (Vac)			Frequency	Current (Amps)
Nominal	Minimum	Maximum	±.5 Hz	Maximum
120	104	127	60 Hz	.65 at 104 Vac

Voltage ac

D.C Outputs

Vdc Voltage	Current (Amps)		Regulation Tolerance
Nominal	Minimum	Maximum	±%
+5	*1.5	3.6	5
+12	.04	1.2	5
-6	0.0	.025	16

Voltage dc

* There must be a minimum of a 1.5 Amp load on the +5 Vdc output for the -6 Vdc to be present.

Over-Voltage/Over-Current Protection

Input (Transformer)

The following table describes the transformer input protection:

Voltage (Nominal)	Type Protection	Rating (Amps)
120 Vac	Non-resettable Fuse Thermal/Over-Current	5A Slo Blow

Input Protection

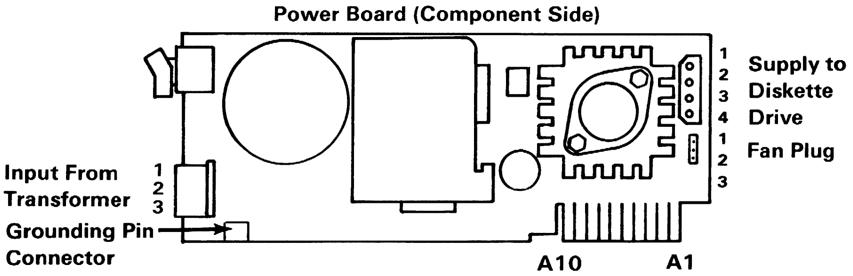
Output (Power Board)

The following table describes the Power Board's output protection:

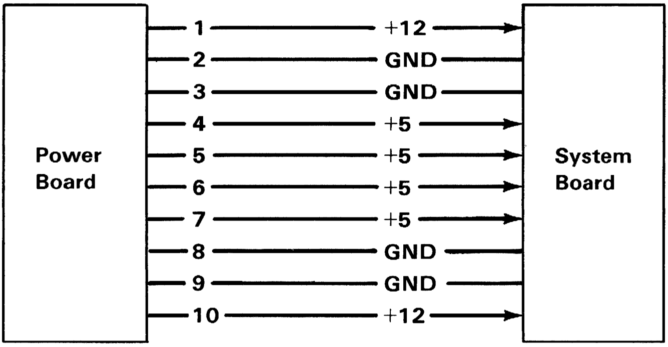
	Protection Condition	
Output Voltages	Over-Voltage	Over-Current
+5 Vdc	*6.3 ± .7 Vdc	**3.9 ± .25 Amps
12 Vdc	*14.4 ± 1.4 Vdc	2.2 ± .9 Amps

* Over-Voltage protection is provided by fuse F1.
**Resettable by removing the fault condition and removing power for at least 5 seconds and then applying power.

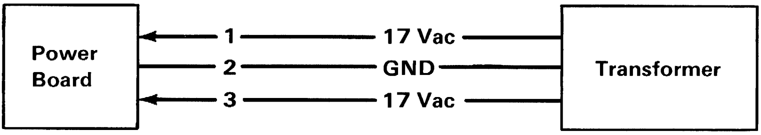
Output Protection



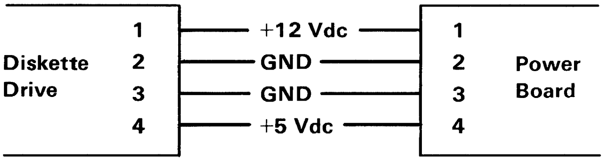
Connector Specifications



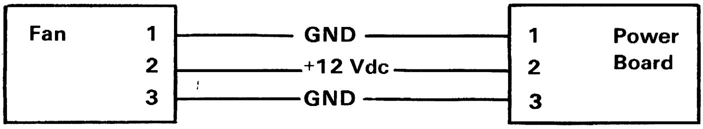
Connector Specifications



Connector Specifications



Connector Specifications



Fan Connector Specifications

Notes:

SECTION 3. SYSTEM OPTIONS

Contents

Notes:

IBM PC*jr* 64KB Memory and Display Expansion

The 64KB Memory and Display Expansion option enables the user to work with the higher density video modes while increasing the system's memory size by 64K bytes to a total of 128K bytes. The memory expansion option plugs into the 44-pin memory expansion connector on the system board. Only one memory expansion is supported.

The Memory Expansion Option does not require the user to reconfigure the system to recognize the additional memory.

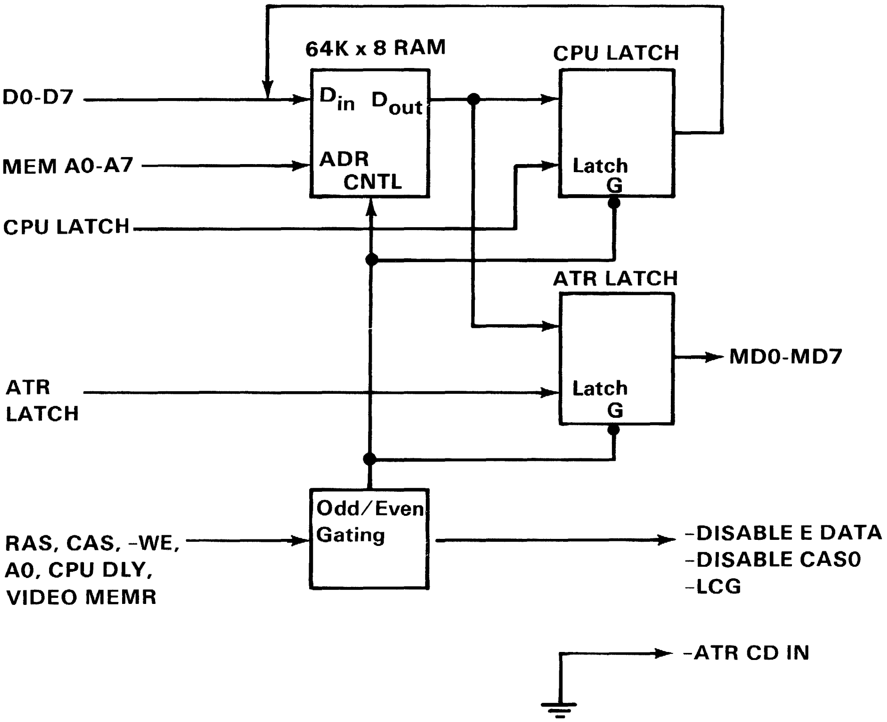
Eight 64K-by-1, 150 ns, dynamic memory modules provide 64K bytes of storage. The memory modules are Motorola's MCM6665AL15, and Texas Instrument's TMS4164-15, or equivalent.

When inserted, the memory expansion option uses the ODD memory space, while the system memory is decoded as the EVEN memory. Thus, when used as video memory, the memory expansion option has the video attributes while the on-board system memory has the video characters. This arrangement provides a higher bandwidth of video characters.

In addition to the eight memory modules, the expansion card has logic to do the EVEN/ODD address decoding, video data multiplexing, and a CARD PRESENT wrap.

Dynamic-refresh timing and address generation are done on the system board and used by the memory expansion option.

The following is a block diagram of the IBM PC*jr* 64KB Memory and Display Expansion.



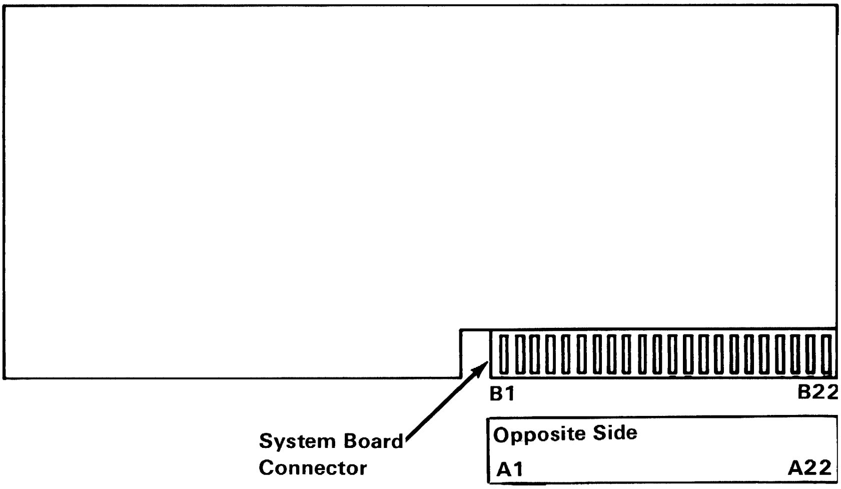
Memory Expansion Block Diagram

Signal	I/O	Description
+RAS	I	+Row Address Strobe. This line is inverted and then becomes the -RAS for the RAM modules.
+A0	I	Microprocessor Address 0. This is used to determine whether the microprocessor access is from the system board RAM (Low) or from the expansion RAM (High).
-DISABLE EDATA	O	When the expansion RAM card is in and the microprocessor is reading an ODD byte of data the expansion card tri-states the latch for EVEN data on the system board using this line.
ATR LATCH	I	This signal indicates that the expansion RAM card should 'latch' up data from the expansion RAM into the attribute latch.
MD0 thru MD7	O	These data lines contain CRT information from the attribute latch and go to the Video Gate Array.
D0 thru D7	I/O	These data lines are from the microprocessor and are bidirectional.
MEM A0 thru A7	I	These are the multiplexed address lines for the dynamic-RAM modules. These lines are multiplexed between row address and column

		address, and also between microprocessor and CRT addresses.
VIDEO MEMR	I	When this signal is 'high' it indicates a MEMR is accessing the system board or expansion RAM is being accessed. This line along with A0 determines if the expansion RAM microprocessor latch should 'gate' its data onto the D0 thru D7 Bus.
CPU DLY	I	This line when 'high' indicates that a microprocessor RAM cycle is occurring. It is used to gate 'off' the expansion RAM CAS or used with A0 to generate the -DISABLE CAS 0 signal.
-DISABLE CAS 0	O	This line is used to disable the system board CAS0 when a system microprocessor 'write' is occurring to the expansion RAM. This line keeps the 'write' from occurring to the system board RAM.
+CAS	I	Column Address Strobe. This line instructs the expansion RAM to 'latch' up the address on the MEM A0 thru A7 address lines.

-LCG	O	This line is used to instruct the system board that attributes or ODD graphics data should be 'read' from the expansion RAM card for use by the Video Gate Array.
GATE	I	This line is 'wrapped' and becomes the -LCG output.
-WE	I	This line instructs the memory that the cycle is a microprocessor 'write' cycle.
CPU LATCH	I	This line instructs the expansion RAM card to 'latch' the data from the expansion RAM into the microprocessor latch.
-ATR CD IN	O	This line is a wrap of the ground line on the expansion RAM card. It pulls 'down' an 8255 input so that the microprocessor can tell if this card is installed or not.

The following is the connector specifications for the IBM PC*jr* 64KB Memory and Display Expansion.



64KB Memory and Display Expansion

Connector Pin	Signal Name	Signal Name	Connector Pin
A01	+RAS	VIDEO MEMR	B01
A02	A0	CPU DLY	B02
A03	-DISABLE EDATA	-DISABLE CAS 0	B03
A04	ATR LATCH	+CAS	B04
A05	MD4	-LCG	B05
A06	MD5	GATE	B06
A07	MD6	Ground	B07
A08	MD7	Ground	B08
A09	MD0	Ground	B09
A10	MD1	-WE	B10
A11	MD2	CPU LATCH	B11
A12	MD3	-ATR CD IN	B12
A13	GND	GND	B13
A14	VCC	VCC	B14
A15	D7	D6	B15
A16	D5	D4	B16
A17	D3	D2	B17
A18	D1	D0	B18
A19	MEM A6	MEM A7	B19
A20	MEM A4	MEM A5	B20
A21	MEM A2	MEM A3	B21
A22	MEM A0	MEM A1	B22

Connector Specifications

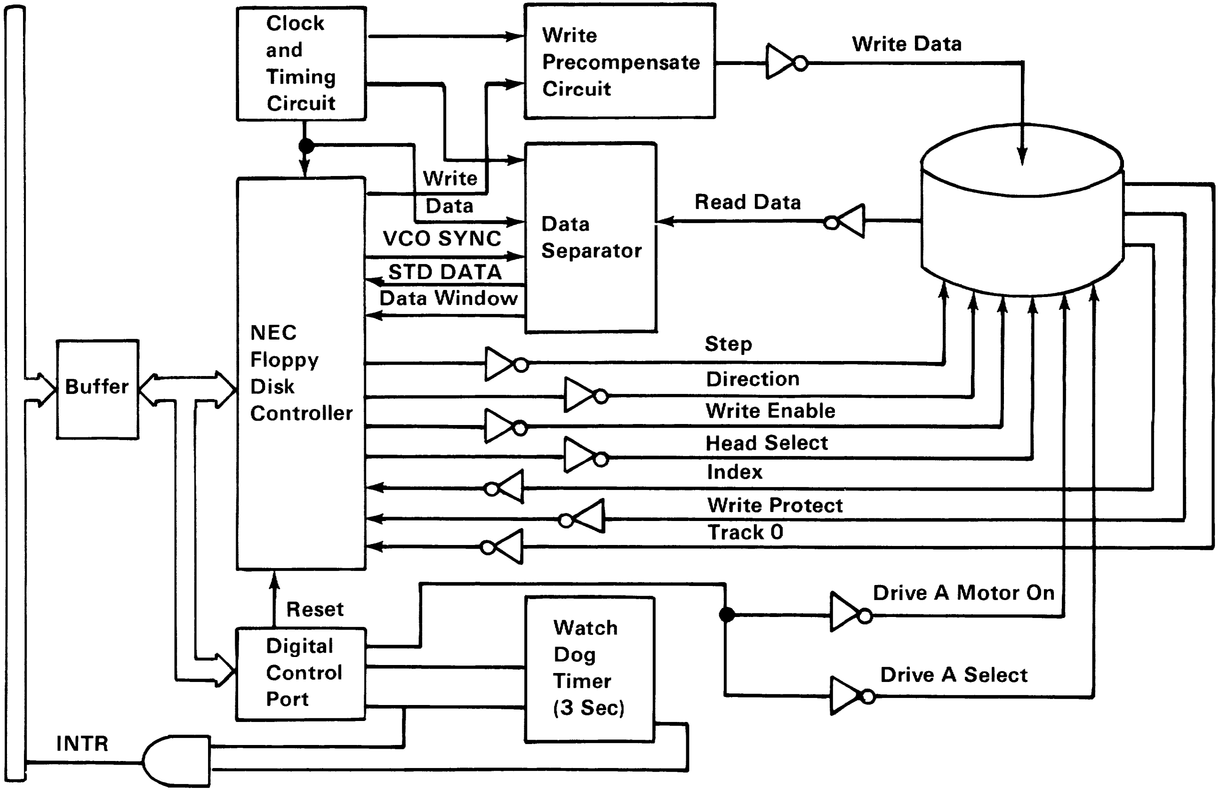
Notes:

IBM PC*jr* Diskette Drive Adapter

The diskette drive adapter resides in a dedicated connector on the IBM PC*jr* system board. It is attached to the single diskette drive through a flat, internal, 60-conductor, signal cable.

The general purpose adapter is designed for a double-density , Modified Frequency Modulation (MFM)-coded, diskette drive and uses write precompensation with an analog phase-lock loop for clock and data recovery. The adapter uses the NEC μPD765 or compatible controller, so the μPD765 characteristics of the diskette drive can be programmed. In addition, the attachment supports the diskette drive's write-protect feature. The adapter is buffered on the I/O bus and uses the system ROM BIOS for transferring record data. An interrupt level is also used to indicate an error status condition that requires processor attention.

A block diagram of the diskette drive adapter follows.



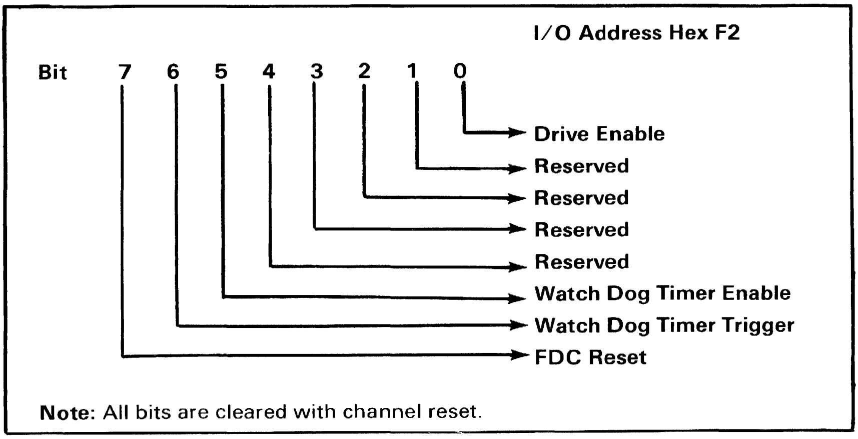
Diskette Drive Adapter Block Diagram

Functional Description

From a programming point of view, the diskette drive adapter consists of a 4-bit digital output register (DOR) in parallel with a NEC μPD765 or equivalent floppy disk controller (FDC).

Digital Output Register

The digital output register (DOR) is an output-only register used to control the drive motor and selection. All bits are cleared by the I/O interface reset line. The bits have the following functions:



Note: All bits are cleared with channel reset.

Digital Output Register

Bit 0 This bit controls the motor and enable lines to the drive. When 'high' (1), this bit will turn 'on' the drive motor and 'enable' the drive. When 'low' (0), this bit will turn 'off' the drive motor and 'disable' the drive.

Bits 1-4 These bits are reserved.

Bit 5 When 'high' (1), this bit 'enables' the WatchDog Timer function and interrupt. When 'low' (0), this bit 'disables' the WatchDog Timer and interrupt.

Bit 6 This bit controls the start of a watchdog timer cycle. Two output commands are required to operate the trigger. A 1 and then a 0 must be written in succession to 'strobe' the trigger.

Bit 7 This bit is the hardware 'reset' for the floppy diskette controller chip. When 'low' (0), this bit holds the FDC in its 'reset' state. When 'high' (1), this bit releases the 'reset' state on the FDC.

WatchDog Timer

The WatchDog Timer (WDT) is a one to three-second timer connected to interrupt request line 6 (IRQ6) of the 8259. This timer breaks the program out of data transfer loops in the event of a hardware malfunction. The WatchDog Timer starts its cycle when 'triggered.'

Floppy Disk Controller (FDC)

The floppy disk controller (FDC) contains two registers that can be accessed by the system microprocessor: a status register and a data register. The 8-bit main-status register contains the status information of the FDC and can be accessed at any time. The 8-bit data register consists of several registers in a stack with only one register presented to the data bus at a time. The data register stores data, commands, parameters, and provides floppy disk drive (FDD) status information. Data bytes are read from or written to the data register in order to program or obtain results after a particular command. The main status register can only be read and is used to facilitate the transfer of data between the system microprocessor and FDC.

FDC Register	I/O Address
Data Register	hex F5
Main Status Register	hex F4

Programming Summary

The FDC is set up with the following Parameters during system power up:

Parameter	Power-up Condition
Sector Size	hex 02 for 512 Byte Sectors
Sector Count	9
Head Unload	hex 0F - Has no effect on system operation.
Head Step Rate	hex D - This gives a step rate of 6 milliseconds.
Head Load Time	hex 1 Minimum head load time.
Format Gap	hex 50
Write Gap	hex 2A
Non-DMA Mode	hex 1
Fill byte for Format	hex F6

FDC Power-up Parameters Settings

The IBM PC*jr* Diskette Drive Adapter and BIOS use and support the following FDC commands:

- Specify
- Recalibrate
- Seek
- Sense interrupt status
- Sense Drive status
- Read data
- Write data
- Format a track

Note: Please refer to the Diskette section of the BIOS listing for details of how these commands are used.

The following FDC hardware functions are not implemented or supported by the IBM PC*jr* Diskette Drive Adapter.

- DMA data transfer
- FDC interrupt
- Drive polling and overlapped seek
- FM data incoding
- Unit select status bits

2 Heads (1 per side)
40 Cylinders (Tracks)/Side
9 Sectors/Track
512 Bytes/Sector
Modified Frequency Modulation (MFM)

Diskette Format

Constant	Value
Head Load	Not Applicable
Head Settle	21 Milliseconds
Motor Start	500 Milliseconds

Drive Constants

Comments

1. Head loads when diskette is clamped.
2. Following access, wait Head Settle time before RD/WR.
3. Drive motor should be 'off' when not in use. Wait Motor Start time before RD/WR.
4. All system interrupts except IRQ6 must be 'disabled' during diskette data transfer in order to prevent data under-run or over-run conditions from occurring.

System I/O Channel Interface

All signals are TTL-compatible:

Most-Positive Up-Level	+ 5.5 Vdc
Least-Positive Up-Level	+ 2.7 Vdc
Most-Positive Down-Level	+ 0.5 Vdc
Least-Positive Down-Level	- 0.5 Vdc

The following lines are used by this adapter:

+D0 thru 7	(Bidirectional, Load: 1 74LS, Driver: 74LS 3-state)

These eight lines form a bus through which all commands, status, and data are transferred. Bit 0 is the low-order bit.

+A0 thru 3 (Adapter Input, Load: 1 74LS)

These four lines form an address bus by which a register is selected to receive or supply the byte transferred through lines D0-7. Bit 0 is the low-order bit.

-IOW (Adapter Input, Load: 1 74LS)

The content of lines D0-7 is stored in the register addressed by lines A0-3 at the trailing edge of this signal.

-IOR (Adapter Input, Load: 1 74LS)

The content of the register addressed by lines A0-3 is 'gated' onto lines D0-7 when this line is 'active.'

-RESET (Adapter Input, Load: 1 74LS)

A down level 'aborts' any operation in process and 'clears' the digital output register (DOR).

+IRQ6 (Adapter Output, Driver: 74LS 3-state)

This line is made 'active' when the WatchDog timer times out.

-DISKETTE CARD INSTALLED (Adapter Output, Driver: Gnd.)

This line is pulled 'up' on the System Board and is wired to input port bit PC2 on port hex 62 of the

8255. This line is used by the program to determine if the diskette drive adapter is installed.

-Diskette CS (Adapter Input, Load: 1 74LS)

This line is shared with the modem CS line and is 'low' whenever the microprocessor is doing IOR or IOW to either the diskette adapter or the modem. This line should be conditioned with A9 being 'low' to generate a DISKETTE CS.

A9 (Adapter Input, Load: 1 74LS)

This line is the microprocessor address line 9. When this line is 'low' and -DISKETTE CS is 'low', IOR and IOW are used by the diskette adapter.

DRQ 0 (adapter Output, Driver: NEC μpd 765)

This output would indicate to a DMA device on the external I/O Channel that the diskette controller wants to 'receive' or 'transmit' a byte of data to or from memory.

DACK 0 (Adapter input, Load: NEC μpd 765)

This line should come from an external DMA and should indicate that a byte is being transferred from/to the Floppy Disk Controller to/from memory.

Drive Interface

All signals are TTL-compatible:

Most Positive Up Level	+ 5.5 Vdc
Least Positive Up Level	+ 2.4 Vdc
Most Positive Down Level	+ 0.4 Vdc
Least Positive Down Level	- 0.5 Vdc

All adapter outputs are driven by active collector gates. The drive should not provide termination networks to Vcc (except Drive Select which has a 2,000 ohm resistor to Vcc).

Each attachment input is terminated with a 2,000 ohm resistor to Vcc.

Adapter Outputs

-Drive Select (Driver: MC3487)

This line is used to 'degate' all drivers to the adapter and receivers from the adapter (except Motor Enable) when the line is not 'active.'

-Motor Enable (Driver: 74LS04)

The drive must control its spindle motor to 'start' when the line becomes 'active' and 'stop' when the line becomes 'inactive.'

-Step (Driver: MC3487)

The selected drive must move the read/write head one cylinder in or out as instructed by the Direction line for each pulse present on this line.

-Direction (Driver: MC3487)

For each recognized pulse of the step line the read/write head should move one cylinder toward the spindle if this line is active, and away from the spindle if not-active.

-Write Data (Driver: 74LS04)

For each 'inactive' to 'active' transition of this line while Write Enable is 'active', the selected drive must cause a flux change to be stored on the diskette.

-Write Enable (Driver: MC3487)

The drive must 'disable' write current in the head unless this line is 'active.'

-HEAD SELECT 1 (Driver: MC3487)

This interface signal defines which side of a two-sided diskette is used for data recording or retrieval. A 'high' level on this line selects the R/W head on the side 1 surface of the diskette. When switching from side 0 to side 1 and conversely, a 100 μs delay is required before any 'read' or 'write' operation can be initiated.

Adapter Inputs

-Index The selected drive must supply one pulse per diskette revolution on this line.

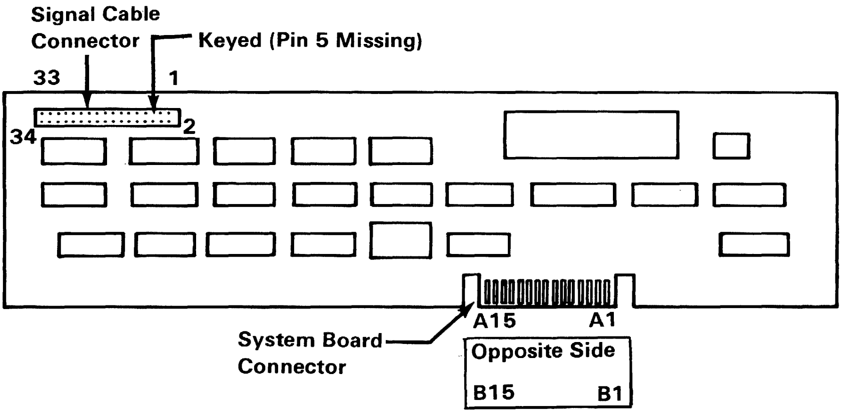
-Write Protect The selected drive must make this line 'active' if a write-protected diskette is mounted in the drive.

-Track 0 The selected drive must make this line 'active' if the read/write head is over track 0.

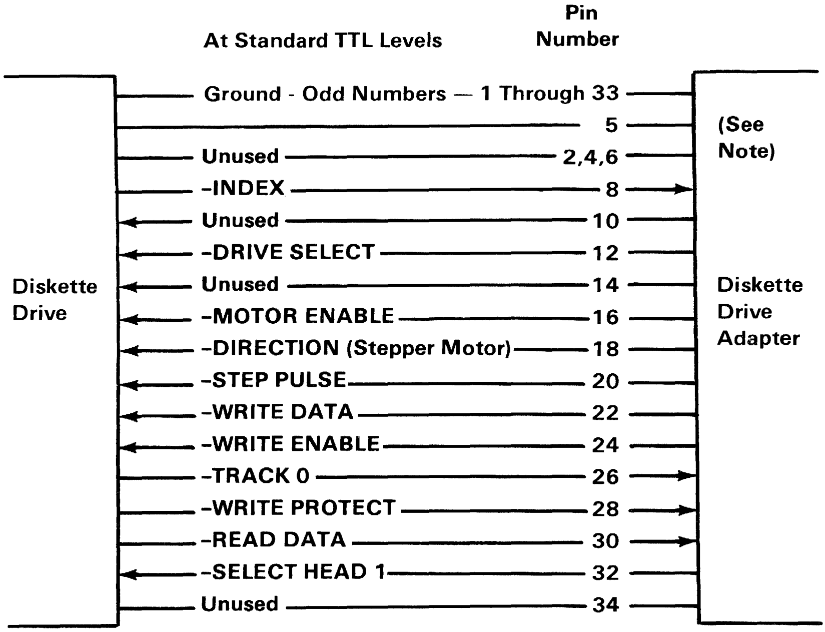
-Read Data The selected drive must supply a pulse on this line for each flux change encountered on the diskette.

Voltage and Current Requirements

The diskette drive adapter requires a voltage supply of +5 Vdc +/- 5% and draws a nominal current of 525 mA and a maximum current of 700 mA.



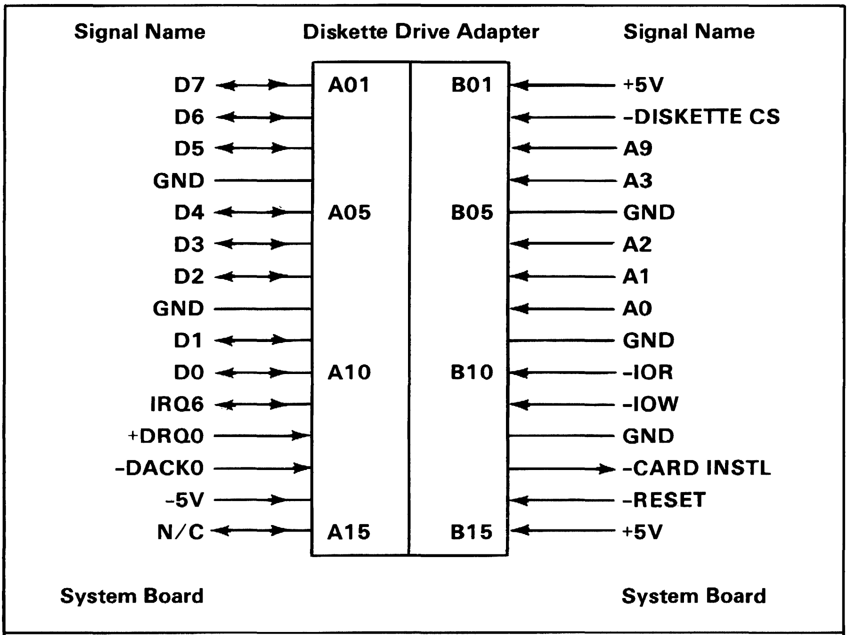
Diskette Drive Adapter



	At Standard TTL Levels	Pin Number	
	Ground - Odd Numbers —	1 Through 33	
		5	
	Unused	2,4,6	
	-INDEX	8	→
←	Unused	10	
←	-DRIVE SELECT	12	
←	Unused	14	
←	-MOTOR ENABLE	16	
←	-DIRECTION (Stepper Motor)	18	
←	-STEP PULSE	20	
←	-WRITE DATA	22	
←	-WRITE ENABLE	24	
	-TRACK 0	26	→
	-WRITE PROTECT	28	→
	-READ DATA	30	→
←	-SELECT HEAD 1	32	
	Unused	34	

Note: Pin 5 is missing to match the key plug on the signal cable.

Connector Specifications (Part 1 of 2)



Note: All levels are TTL compatible.

Connector Specifications (Part 2 of 2)

IBM PC*jr* Diskette Drive

The system unit has space and power for one diskette drive. The drive is double-sided with 40 tracks for each side, is fully self-contained, and consists of a spindle-drive system, a read- positioning system, and a read/write/erase system.

Functional Description

The diskette drive uses modified frequency modulation (MFM) to read and write digital-data, with a track-to-track access time of 6 milliseconds.

To load a diskette, the operator rotates the load lever at the front of the diskette drive clockwise and inserts the diskette into the slot. Plastic guides in the slot ensure the diskette is in the correct position. Closing the load lever centers the diskette and clamps it to the drive hub. This same action also loads the Read/Write heads against the surfaces of the diskette. The load lever is mechanically interlocked to prevent closing of the lever if a diskette is not installed.

The head-positioning system moves the magnetic head to come in contact with the desired track of the diskette. Operator intervention is not required during normal operation. If the diskette is write-protected, a write-protect sensor 'disables' the drive's circuitry, and an appropriate signal is sent to the interface.

Data is read from the diskette by the data-recovery circuitry, which consists of a low-level read-amplifier, differentiator, zero-crossing detector, and digitizing circuits. All data decoding is done by the adapter card.

The IBM PC*jr* Diskette Drive is equipped with a media cooling fan, which gets its power from the power supply board.

The diskette drive also has the following sensor systems:

- The track 00 sensor, senses when the head/carriage assembly is at track 00.
- The index sensor, which consists of an LED light source and phototransistor. This sensor is positioned so that when an index hole is detected, a digital signal is generated.
- The write-protect sensor 'disables' the diskette drive's electronics whenever it senses a write-protect tab on the diskette.

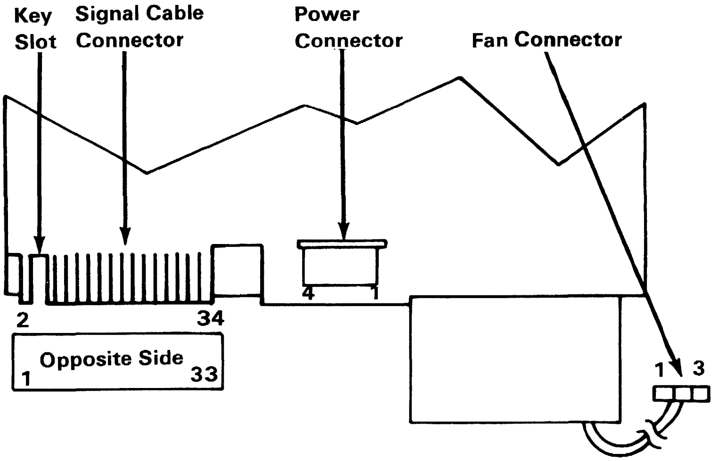
The drive requires power within the following specifications:

Specification	+5 Vdc Input	+12 Vdc Input
Nominal Supply	+5 Vdc	+12 Vdc
Ripple (0 to 50 kHz)	100 mV	100 mV
Tolerance (Including Ripple)	±5%	±5%
Standby Current (Nominal)	600 mA	400 mA
Standby Current (Worst Case)	700 mA	500 mA
Operating Current (Nominal)	600 mA	900 mA
Operating Current (Worst Case)	700 mA	2400 mA

Diskette Drive Power Specifications

For interface information refer to "Diskette Drive Adapter" in this section.

For mechanical and electrical specifications see Appendix D.



Diskette Drive Connectors

Diskette Drive		Power Board
1	+12 Vdc	1
2	GND	2
3	GND	3
4	+5 Vdc	4

Connector Specifications (Part 1 of 2)

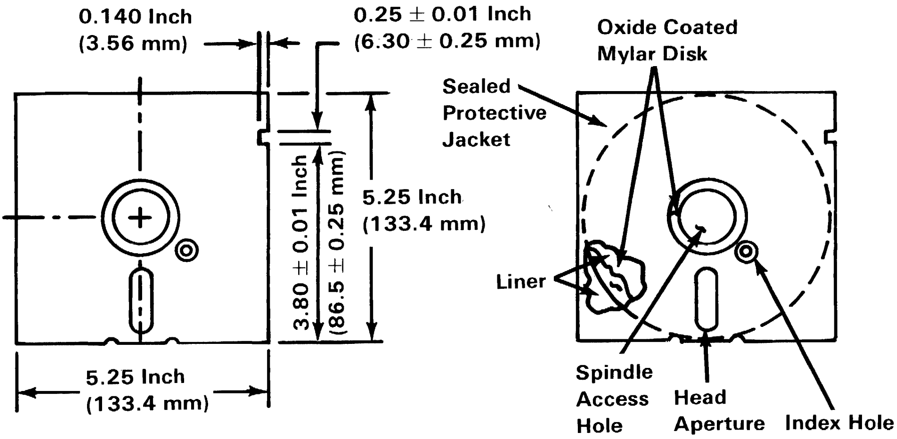
Diskette Drive Fan		Power Board
1	GND	1
2	+12 Vdc	2
3	GND	3

Connector Specifications (Part 2 of 2)

Notes:

Diskette

The IBM PC*jr* Diskette Drive uses a standard 133.4 mm (5.25 in.) diskette. For programming considerations, single-sided, double-density, soft-sectored diskettes are used for single-sided drives. Double-sided drives use double-sided, double-density, soft-sectored diskettes. The figure below is a simplified drawing of the diskette used with the diskette drive. This recording medium is a flexible magnetic disk enclosed in a protective jacket. The protected disk, free to rotate within the jacket, is continuously cleaned by the soft fabric lining of the jacket during normal operation. Read/write/erase head access is through an opening in the jacket. Openings for the drive hub and diskette index hole are also provided.



Recording Medium

Notes:

IBM PC*jr* Internal Modem

The IBM PC*jr* Internal Modem is a 65 mm (2.5 inch) by 190 mm (7.5 inch) adapter that plugs into the PC*jr* system board modem connector. The modem connector is an extension of the system I/O bus. All system control signals and voltage requirements are provided through a 2 by 15 position card-edge tab with 0.254 cm (0.100-inch) spacing on the modem adapter.

Functional Description

The Internal Modem consists of two major parts: (1) the INS8250A Asynchronous Communication Element, and (2) the Smart 103 Modem. Therefore, the programming must be considered in two parts. The INS8250A communications protocol is a function of the system ROM BIOS, and is discussed later in this section. All 'pacing' of the interface and control-signal status must be handled by the system software. After the INS8250A is initialized, the modem is controlled by ASCII characters transmitted by the INS8250A.

Key features of the INS8250A used in the modem adapter are:

- Adds or deletes start bits, stop bits, and parity bits to or from the serial data stream
- Full double-buffering eliminates the need for precise synchronization
- Independently-controlled transmit, receive, line status, and data-set interrupts
- Programmable baud-rate-generator allows division of the baud clock by 373 (hex 175) for a 300-bps transmission-speed or 1017 (hex 3F9) for a 110-bps transmission-speed to generate the internal 16 x clock

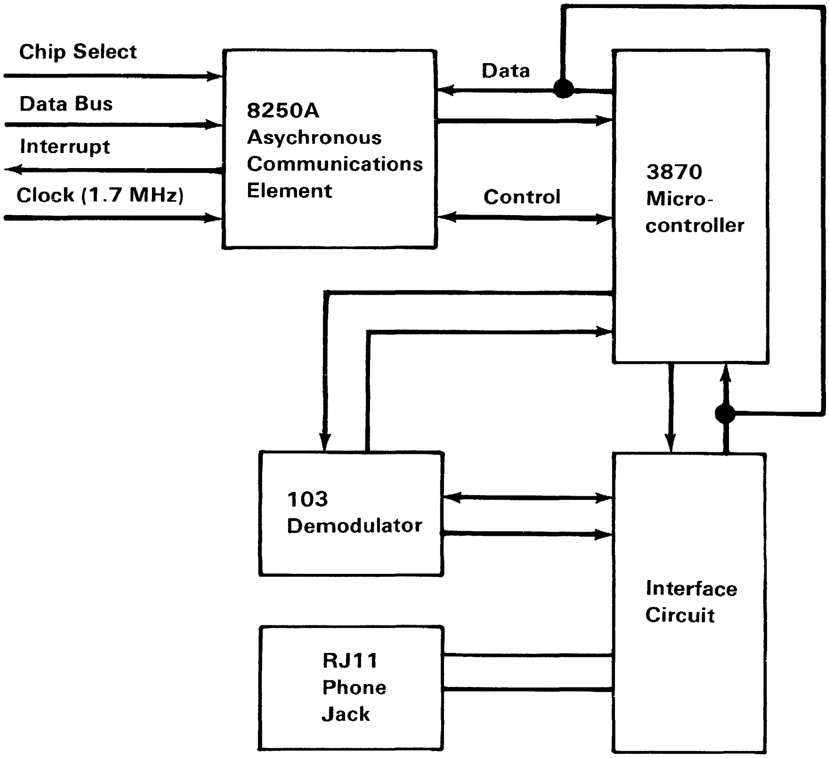
- Modem-control functions: Clear to Send (CTS), Data Set Ready (DSR), Data Terminal Ready (DTR), Ring Indicator (RI), and Data Carrier Detect (DCD)
- Fully-programmable serial-interface

  characteristics:
  - 7, or 8-bit characters
  - Even, odd, or no-parity bit generation and detection
  - 1 stop-bit generation
  - Baud-rate generation
- False-start bit detection
- Complete status reporting capabilities
- Line-break generation and detection
- Internal-diagnostic capabilities
  - Loopback controls for communications-link fault-isolation
  - Break, parity, overrun, framing-error simulation
- Fully prioritized-interrupt system-controls

Key features of the Smart 103 Modem used on the IBM PC*jr* Internal Modem are:

- Direct connection to a telephone company line through an FCC Part-68-approved permissive connection
- Compatible to Bell Series 100 originate/answer for modulation and handshaking
- All functions controlled by ASCII characters and INS8250A modem-control lines
- Uses modular phone-jack (USOC RJ11)
- Data rate is either 300 or 110 bits-per-second
- Auto/manual originate
- Auto/manual answer
- Communication mode is full duplex on two-wire, switched-network channels

- Auto dialer; either DTMF ([dual-tone modulated-frequency] touch-tone) or pulse-dialing (rotary dial) by software command
- Tandem dialing
- Call-progress reporting
- Dial-tone, ring-back tone, and busy-tone detection



IBM PC*jr* Internal Modem Block Diagram

Modem Design Parameters

The following tables describe the design parameters of the Smart 103 Modem.

Dialer Type:	Two modes 1. Forced Touch-Tone (DTMF) dialing 2. Forced pulse dialing
Tandem Dialing:	The ASCII character P (hex 50 or 70) in the dial string causes a delay of up to 10 seconds while the modem is searching for another dial tone. A time out will cause the modem to hang up and post status. The ASCII character W (hex 57 or 77) in the dial string causes a 5-second dead wait before continuing to dial. Multiple ASCII W's will cause multiple waits.
Pulse Dialing:	Rate: 10 + 1, –0 pulses per second Duty Cycle: 60% make, 40% break Interdigit Delay: 800 ms ± 50 ms
DTMF Dialing:	Tone Duration: 85 ms ± 10 ms Intertone Duration: 80 ms ± 10 ms

Dialer Parameters (Part 1 of 2)

Tone Pair Frequencies:		
ASCII Digit Code	Frequency (Hz)	
0	941	1336
1	697	1209
2	697	1336
3	697	1477
4	770	1209
5	770	1336
6	770	1477
7	852	1209
8	852	1336
9	852	1477
*	941	1209
#	941	1477

Dialer Parameters (Part 2 of 2)

Time Out Duration: A data call will time out if an answer tone is not detected within 45 seconds of the last digit dialed.

Failed Call Time Out Parameter

Modulation: Conforms to Bell 103/113 specification using binary phase-coherent frequency shift keying (FSK).

Modulation Parameter

Mode	Originating End	Answering End
Transmit	1070 Space 1270 Mark	2025 Space 2225 Mark
Receive	2025 Hz Space 2225 Hz Mark	1070 Hz Space 1270 Hz Mark

Transmitter/Receiver Frequency Parameters

Receive Sensitivity	More negative or equal to -42 dBm.

Receive Sensitivity Parameters

Transmitter Level	Fixed at -10 dBm as per FCC Part 68 Permissive connection.

Transmitter Level Parameter

Programming Considerations

The modem and the IBM PC*jr* system can communicate commands or data between each other. Any commands sent to the modem from the IBM PC*jr* are stripped from the data stream and executed but are not transmitted to the receiving station. The data is transparent to the modem. The modem is capable of causing hardware interrupts as the result of certain conditions, and in response to queries for its status.

Commands to the modem are a sequence of characters preceded by a single command character. The command character tells the modem that the following character sequence, until a carriage return, is a command. The carriage return completes the command sequence and causes the modem to execute the commands. The command character (represented by **[cc]** in the following text) is programmable (with the **NEW** command) to any ASCII character (hex 00 thru 7F). The default for the command character is Ctrl N (ASCII hex 0E).

Commands can occur anywhere in the data stream if properly formatted but are not to be executed by the modem until a carriage return is received.

Multiple commands are allowed if separated by commas and preceded by a single command character.

Command Format

The following is the command format that all commands must follow.

[cc][command word][delimiter][arguments] [,more][CR]

where:

[cc]	is the single ASCII command character.
[command word]	is the command word or the first letter of the command word.
[delimiter]	is always a space when separating an argument and command word. Any spaces thereafter are ignored until the modem sees a comma, an argument or a carriage return.
[arguments]	is a variable that is replaced by any character allowed by the command definition.
[,more]	is any additional commands preceded by a comma.
[CR]	is a carriage return that completes the command sequence and causes the modem to execute the commands.

The following are two examples of command format.

[cc] COUNT 5 [CR]
sample test **[cc] VOICE, D** (408)
555-1234,**QUERY [CR]**

Format Guidelines

1. Commands can occur anywhere in the data stream if properly formatted but are not be executed by the modem until a carriage return is received.
2. Multiple commands are allowed if separated by commas and preceded by a single command-character.
3. Only the first character of the command word is significant. All remaining characters are ignored up to the first space following the command word. In other words, the **DIAL** command and **DUMMY** are treated identically.

4. The modem does not discriminate between upper-case and lower-case characters.
5. There are three ways to send the current command-character as data to a receiving station:
   a. Consecutively sending it twice:

      [cc][cc]

      This would send the character a single time.
   b. Change the command character (with the **NEW** command) to another ASCII character and then transmit the previous command-character.
   c. Place the modem in the Transparent mode and then transmit the character.

Commands

The commands that are used with the integrated modem are listed on the following pages in alphabetical order.

Each of the commands has its syntax described according to the following conventions:

1. Words in capital letters are keywords. Only the first letter of the keyword is required, the others are optional.

2. You must supply any arguments which are in lower-case letters. Valid characters for arguments are defined as:

   - m - ASCII decimal digits 0 to 9, *, #, I, P, and W

   - n - ASCII hexadecimal digits 0 to F

   - o - ASCII hexadecimal digits 0 to 9

   - p - any ASCII character

3. All arguments are examined for validity. If extra characters are used in an argument, the extra characters are ignored. If the argument is invalid, the command is ignored.

4. An ellipsis (...) indicates an item may be repeated as many times as you wish.

5. All command lines must begin with a command character. The default command-character is (CONTROL N).

6. Multiple commands separated by commas can follow a single command-character.

An example of the **DIAL** command is given below:

Command format - **DIAL m...m**

Command line - DIAL 1 800 555 1234

If an invalid argument or no argument is given, the command is not executed. Also, a question mark (?) is given as the error response and the command line is aborted.
The commands are as follows:

Format: **ANSWER**

A

Purpose: To logically take the phone off the hook and force **ANSWER** mode. This is logically like a manual answer.

Format: **Break n**

Purpose: To send a space or break character for a duration equal to a multiple of 100 ms (**n** x 100 ms).

Format: **COUNT n**

C n

Where **n** is the number of complete rings in the range of hex 0 to hex F.

When answering an incoming call, the modem answers the phone after **n** complete incoming rings, where **n** is any value from hex 0 to F.

A value of zero specifies that the modem not answer an incoming call, but still carry out any instructions from the host.

When dialing, the modem waits **n** + 3 complete ringbacks before cancelling the call.

If **n** exceeds 4, the 45-second abort timer cancels an outgoing call with an "UNSUCCESSFUL" response, as more than seven ringbacks exceeds 45 seconds.

Purpose: Sets the ring count when the modem is answering an incoming call or dialing a call.

Default: 0

Format: **DIAL m...m**

D m...m

Where **m...m** is a dial string of ASCII decimal digits 0 through 9, *, #, I, P, and W. A maximum of 33 characters are allowed in the dial string. The first character of the string defaults to P (a 10-second delay while searching for the dial tone). W causes the modem to delay five seconds, then continue dialing.

W or P must start a string, can also occur anywhere within a string, and causes the digits to be tone dialed.

The characters * and # represent the two extra buttons on a push-button phone, but may be used for other things.

I causes the next digits to be pulse dialed. The I stays in effect until a (P,), (W,),or end of command. The modem then searches for line busy, ringing, or incoming carriers while posting the status.

Purpose: To cause the modem to dial.

Default: P (10-second timeout). (If this command is used without an argument, the last number dialed is redialed once.)

Format: **FORMAT n**

F n

Where **n** is one of the following:

n	**Parity**	**Data Length**	**Stop Bit**
0	Mark	7	1
1	Space	7	1
2	Odd	7	1
3	Even	7	1
4	None	8	1
5-7	Reserved		

The 8250A line control register (LCR) must specify the same format as defined in the **FORMAT n** command to 'enable' data/command communication.

Do not combine this command with any other commands except the **SPEED** command on a single command line.

> **Note:** If programming in BASIC, this command must be used in addition to specifying the same parity and data length in the BASIC 'open' statement.

Purpose: To change the parity and number of stop-bits being transmitted at either end, to a new format.

Default: 3

Format: **HANGUP**

H

Purpose: To perform a clean disconnect and go on-hook. Logically the same as manually hanging up.

Format: **INITIALIZE**

I

This command is executed in 10 seconds and is the same as a cold start. An "OK" response is not returned after execution and the integrity test code in the **QUERY** command is set.

Purpose: Places the modem in the power-up default-state.

Format: **LONG RESPONSE o**

L o

Where **o** is one of the following:

o	**Mode**	**Responses**
0	Verbose	"BUSY" "CONNECTED" "NO ANSWER" "NO DIAL TONE" "OK" "RING" "UNSUCCESSFUL" "?" (Question Mark)
1	Terse (Hex code)	30 31 32 33 34 35 36 37

Note: The dial string is not echoed in the terse mode.

Purpose: Modifies message feedback. Information is posted in the status area.

Default: 0 (Verbose mode)

Format: **MODEM**

M

Purpose: Forces the modem into the data state where the carrier is placed on the telephone line and proper connection-protocols are followed.

This command is equivalent to **ANSWER** if the data state started as autoanswer.

Format: **NEW p**

N p

where **p** is any ASCII character.(hex 0E)

Purpose: Changes the command character to an ASCII character.

Default: Ctrl N (ASCII hex 0E)

Format: **ORIGINATE**

O

Purpose: Logically takes the phone off-hook and forces the ORIGINATE mode. Logically equivalent to manual originate.

Format: **PICKUP**

P

Purpose: Logically takes the phone off-hook and puts the modem in the voice state.

Format: **QUERY**

Q

Purpose: To query the modem for its status information.

Possible characters returned by the modem are as follows:

Responses	**Meaning**
H0 or H1	Hook status: H0 = on-hook, H1 = off-hook.
S0 to SF	Current ringcount setting in hex.
B	Line busy.
D	Line dead: no dial-tone found or no ring/no busy timeout after dialing.
L	Successful dial and handshake.
N	Dial not recorded: dial tone present after dialing.
X	No answer: ringcount plus 3 exceeded.
T0	Integrity test passed.
T1	Integrity test failed.

The first group of characters is always returned for a **QUERY** command. The second group of characters is returned only after a dialing sequence has been started or a change has occurred in the dialing status. The third group of characters is returned when a **TEST** command has occurred. All characters except the first group are erased by being read and do not appear in response to the next **QUERY** unless the condition has recurred in the interim. The **QUERY** response overrides any incoming data from the telephone line.

Format: **RETRY**

R

Purpose: When placed after a **DIAL** command, it causes the modem to execute up to 10 redials at a rate of one per 40 seconds. The redials are triggered by a busy detection after dialing.

Format: **SPEED o**

S o

Where **o** is one of the following:

o	**bps**
0 -	110
1 -	300
2 -	Reserved

> **Note:** Do not combine this command with other commands except the **FORMAT** command on a single command line.

The **SPEED** command must be issued before the 8250A baud rate is changed.

> **Note:** If programming in BASIC, this command must be used in addition to specifying the same bps rate in the BASIC 'open' statement.

Purpose: Sets the baud rate.

Default: 1 (300 bps)

Format: **TRANSPARENT n...n**

T n...n

Where **n...n** is the number of bytes to transmit in the range of hex 0 to hex FFFF.

Purpose: Places the modem in the transparent mode for the next **n...n** bytes.

The modem does not look for command sequences but instead transmits every character it receives.

The argument can be up to four ASCII-coded hex digits long. This provides a range of 65,536 bytes.

If an argument is not included with the **TRANSPARENT** command, the command is ignored because it has no default.

The transparent mode is terminated when:

1. **n...n** characters have been transmitted.
2. Loss of carrier timeout.
3. INS8250A OUT 1 pin goes 'active.' (The INS8250A -OUT 1 signal should remain 'active' until the transparent mode is requested again.)

The modem exits the transparent mode before processing the next complete character from the host.

To re-enter the transparent mode, the sequence is:

1. The INS8250A -OUT 1 pin changes to, or remains in the 'inactive' state.

2. The command string containing the **TRANSPARENT** command is issued.

   An argument of 0 causes a permanent transparent mode which can be exited by the INS8250A -OUT 1 pin going 'active.'

Format: VOICE

V

Purpose: Forces the modem to the voice state where no tones or carriers are placed or searched for on the telephone line.

This state is used for voice communication, when the modem is an autodialer or answering device only. It is also necessary to be in the voice state to transmit DTMF tone-pairs.

This command 'disables' the autoanswer function.

The status responses are:

1. If a busy signal is detected "BUSY OK".

2. Any other condition "OK...(16 dots)....CONNECTED".

Format: **WAIT**

W

Purpose: Causes the modem to take no action, including autoanswer, until the next command is received from the host. All commands following the **WAIT** command in a single command-line are ignored.

Format: **XMIT m...m**

X m...m

Purpose: Instructs the modem to transmit the DTMF tone-pairs found in the argument string **m...m.** This is only valid in the voice state. Delays between digits can be caused by inserting W's in the string.

Each W causes a five-second delay.

Format: **ZTEST o**

Z o

Where **o** is one of the following:

o	**Test**
0 -	Hardware Integrity Test
1 -	Analog Loop Back Test

Purpose: Places the modem in the test mode specified by the argument.

For modes other than the integrity test, the modem stays in the test mode until any other command is received.

For the integrity test, the test is performed, status posted, and then the modem returns to service immediately. The integrity test takes eight to 10 seconds to execute and its completion is signaled by an "OK" message.

All commands following the **ZTEST** command in a single command-line are ignored.

Responses

Autoanswer

If -DTR is 'active', the modem goes off-hook and proper connection protocols including the two-second billing delay are followed. If connection is made, the modem sends "CONNECTED" to the host and posts the status in the status area.

Editing/Changing Command Lines

Corrections to the command line can be performed by aborting current-command lines and typing a new line or by entering the correct command later on in the current-command line.

The last command entered on a single command-line supersedes any previously entered command that performs an opposite function.

A Control X or backspace received by the modem immediately aborts the entire command line.

Opposite Commands

The command line is scanned after its completion (after **[CR]** is entered). Commands which cause an action during the scan (for example, **DIAL**) are not candidates for opposite treatment. Only commands which 'preset' a static condition can be opposites.

They include:

Count (n)	two entries, latest are used
Format (n)	two entries, latest are used
New (p)	two entries, latest are used
Speed (n)	two entries, latest are used
Transparent n..n	two entries, latest are used
Modem - Voice	these are opposites only when on-hook

Note: Answer and originate are not opposites; each of these causes an action when scanned.

Status Conditions

The modem sends the host messages as defined in the **LONG RESPONSE** command for dialing success or failure. Hardware interrupts for carrier loss and detecting incoming rings are provided on the 8250A.

Dialing and Loss of Carrier

The dialing process begins with the modem searching for a dial tone if it is not in the blind dialing mode. If a dial tone is not detected, the modem hangs up, the appropriate status characters are posted, and the "NO DIAL TONE" message is returned to the host.

If a dial tone is found, the modem continues to dial. When a P is encountered in the dial string, the modem delays for up to 10 seconds to search for another dial tone and returns the "NO DIAL TONE" message to the host if a dial tone is not detected. When a W is encountered in the dial string, the modem delays for five seconds before continuing to dial. Consecutive W's are allowed in a dial string.

Anytime a P or W is not followed with an I in a dial string, the next digits are tone-dialed. When an I follows a P or W, all following digits are pulse-dialed until a P, W, or end of command ([**CR**]) is detected.

The modem ignores any character except 0 through 9, *, #, I, P, or W while dialing. This allows the user to place parentheses and dashes in the dial string for greater legibility.

The modem checks the telephone line again after it has dialed the digits in the dial string. If a dial tone is found immediately, the dialed digits are not recorded and the modem posts this to the status characters, hangs up, and sends the "UNSUCCESSFUL" message to the host. If the line is busy, this is also posted to the status characters and the modem hangs up and returns the "BUSY" message to the host. If the line is ringing, the modem begins counting the number of rings. If this count exceeds the value of **COUNT** + 3, the modem hangs up and takes the same actions as above. If no answer tone is detected within 45 seconds after completion of dialing, the modem hangs up and takes the same actions as above.

Finally, if the call is answered, the modem either looks for a carrier and begins the handshake sequence (if it is in the data or modem state) or remains silent (if it is in the voice state). In the voice state, the modem looks for busy, and transmits a response (1) when the line is found not busy, or (2) if it is found busy, in which case it also hangs up and possibly dials again. In voice state, ringback count and abort time out are not used.

If, during the process of establishing the data link after dialing, the modem receives any character from the host or - DTR goes 'inactive', the modem aborts the call with a clean disconnect, clears the balance of the command line, and sends an "OK" message. Also, the modem does not carry out the instruction sent from the host, even if the character is a command character.

In the data state, the modem transmits a message after successful completion of the handshake, or after it has determined that the handshake failed. An unsuccessful handshake is evidenced by absence of carrier at the proper time.

If a carrier drops out for more than two seconds in the data state, the modem begins a timeout lasting approximately 17 seconds. At the end of the timeout, the modem hangs up. Any command received during the 17 seconds resets the timer.

The modem does not automatically reestablish the connection if the carrier returns after this dropout interval. This allows the user or software to intercede by commanding the modem to go into the voice state, to hang up immediately, or to take some other action. The data connection may also be terminated by a **HANGUP** command while carriers are still present. A voice connection is always terminated by a **HANGUP** command.

Default State

Upon power up or after an **INITIALIZE** command is given, the modem returns to the default state as follows:

- A verification of hardware integrity is performed and the result posted to the status characters.
- The remaining status characters cleared.
- The modem is placed in the data state awaiting a dialing request or incoming ring.
- The Transparent mode is cleared.
- All loopback modes are cleared.
- The wait mode is cleared.
- The command character is set to Control-N.
- The data format is set to 7 data bits, even parity, and one stop bit.
- Ringcount is set to 0 (auto answer 'disabled')
- The modem is set to on-hook.
- The message mode is set to verbose.

Programming Examples

Call progress reporting is done in two modes, verbose messages or terse messages as defined in **LONG RESPONSE** command to the Serial In (SIN) pin of the 8250A. The power-up default is the verbose messages mode, and these messages from the modem are in capital letters. Also, in call progress reporting, the status area is updated.

The following examples are representative of real-time call-progress reporting. The italicized entries are user entries.

Example 1:

```
OK [cc]Dial 555-1234 [CR]
NO DIAL TONE
OK
```

In this example, no dial tone is detected within the time out period.

Example 2:

```
OK
[cc]Dial 555-1234 [CR]

5551234.....................
RING ..................................CONNECTED OK
```

In this example, a modem answer tone is detected.

Example 3:

```
OK
[cc]Dial 1(301)555-1234 [CR]
13015551234..... BUSY
OK
```

In this example, busy is detected.

Example 4:

```
OK
[cc]Dial 555-1234 [CR]
5551234........
RING........................
RING.............................
RING..................NO ANSWER
OK
```

In this example, ring count is exceeded before ringing stops.

Example 5:

```
OK
[cc]Dial 555-1234 [CR]
5551234.........
RING.......................................
 ...........................................
 .........................UNSUCCESSFUL
OK
```

In this example, a failed-call time-out occurred because an answer tone was not detected within the allotted time.

Example 6:

```
OK
[cc]Dial 99P555-1234 [CR]
99.......................................................
 ...............NO DIAL TONE
OK
```

In this example, the second dial-tone is not detected within the time out period.

Example 7:

```
OK
[cc]Dial 99P421-7229 [CR]
99............BUSY
OK
```

In this example, busy is detected within the time-out period.

Example 8:

```
OK
[cc]Dial 99WW555-1234 [CR]
99..................................................
 ...................................................
 ...................................................
 ..............
4217229....
RING...................CONNECTED   OK
```

In this example, the access code is dialed and two dead waits are performed. Then, the second number is dialed and a modem answers.

Example 9:

```
OK
[cc]Dial 555-1234, Retry [CR]
5551234..................BUSY
5551234...............BUSY
5551234............CONNECTED   OK
```

In this example, the modem dials a number with auto redial.
The first two times, the number is busy.
The third time, a modem answers.

Modes of Operation

The different modes of operation are selected by programming the 8250A Asynchronous Communication Element. This is done by selecting the I/O address (hex 3F8 to 3FF) and writing data out to the card.

The 8250A is externally programmed to provide asynchronous, ASCII, 10 bit character length including start, stop, and parity on the serial-output pin (SOUT, pin 11). The data rate is 110 or 300 bits-per-second. The commands can be either upper-case or lower-case characters. See the command, **Format [n]**, earlier in this section for additional information.

For further information refer to “Bibliography.”

Hex Address	Register Selected	Input/ Output	Mode		Notes
			1	2	
3F8	Transmit Buffer	Write	XX	XX	*
3F8	Receive Buffer	Read	XX	XX	*
3F8	Divisor Latch LSB	Write	75	F9	**
3F9	Divisor Latch MSB	Write	01	03	**
3F9	Interrupt Enable	Write	0F	0F	*
3FA	Interrupt Identification	Read	XX	XX	
3FB	Line Control	Write	1A	03	
3FC	Modem Control	Write	01	01	
3FD	Line Status	Read	XX	XX	
3FE	Modem Status	Read	XX	XX	
3FF	Scratch Pad	Write	XX	XX	

*DLAB = 0 (Bit 7 in line control Register).
**DLAB = 1 (Bit 7 in line control Register).
Mode 1 - 300 BPS - 7 Data Bits, 1 Stop Bit, Even Parity.
Mode 2 - 110 BPS - 8 Data Bits, 1 Stop Bit, No Parity.

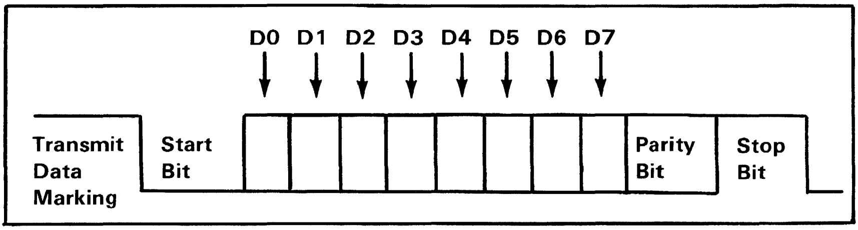
8250A Register Description

Interrupts

One interrupt line is provided to the system. This interrupt is IRQ4 and is 'positive active.' The interrupt enable register must be properly programmed to allow interrupts.

Data Format

The data format is as follows:



Transmitter Output and Receiver Input Data Format

Data bit 0 is the first bit to be transmitted or received. The attachment automatically inserts the start bit, the correct parity-bit if programmed to do so, and the stop bit.

Interfaces

8250A to Modem Interface

The following describes the 8250A to 103 modem interface:

Signal **Description**

INS8250A -OUT 1

The 'inactive' state enables entry into the transparent mode using the **UNLISTEN** command. The 'active' state 'disables' the transparent mode.

-OUT 2

No connection.

SOUT

Serial output from the 8250A.

-RTS

-Request To Send

No connection.

-DTR

-Data Terminal Ready

1. To accept a command, -DTR must be 'active.'

2. If -DTR goes 'inactive', the modem does a clean disconnect sequence.

3. In auto-answer mode, the modem does not go off-hook, but RI on the 8250A will be toggled if the ringing signal is present.

SIN

Serial input to the 8250A.

-RI

The ring indicator pulses with an incoming ring voltage.

-CTS

-Clear To Send

This line is wired 'active' on the modem adapter.

-DSR -Data Set Ready

This line is wired 'active' on the modem adapter.

-RLSD -Received Line Signal Detect

When 'low', this line indicates the data carrier has been detected. If the carrier drops out for longer than two seconds, this line goes 'inactive' and starts the timeout timer.

-RESET, +XRESET These lines are used to reset or initialize the modem logic upon power-up. These lines are synchronized to the falling edge of the clock. Its duration upon power up is 26.5 ms -RESET is 'active low'. +XRESET is 'active high.'

A0,A1,A2,A9 Address bits 0 to 3 and bit 9. These bits are used with -MODEM CS to select a register on the modem card.

-MODEM CS
DISKETTE CS This line is 'active' for addresses hex 0F0 thru 0FF and 3F8 thru 3FF. It is gated with A9 in the 8250A to exclusively decode hex 3F8 thru 3FF.

D0 thru D7 Data bits 0 thru 7:

These eight lines form a bus through which all data is transferred. Bit 0 is the least significant bit (LSB).

-IOR The content of the register addresses by line A0 thru A2 is gated onto lines D0 thru D7 when this line is 'active', -MODEM CS is 'active', and A9 is 'high.'

-IOW The content of lines D0 thru S7 is stored in the register addressed by A0 thru A2 at the leading edge of this signal when -MODEM CS is 'active', and A9 is 'high.'

BAUDCLK This is a 1.7895 MHz clock signal used to drive the Baud Rate Generator.

+MODEM INTR This line is connected to the +IQRP4 on the 8259A Interrupt Controller.

-CARD INSTALL This line indicates to the system BIOS that an IBM PC*jr* Internal Modem is installed in the feature location.

Telephone Company Interface

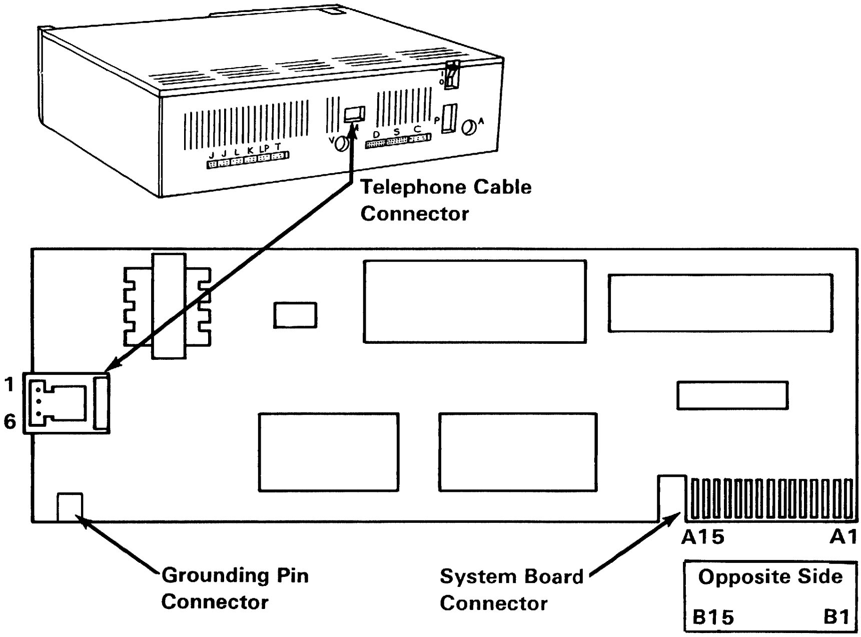
The telephone company interface is a 600 Ohm, balanced, two-wire telephone-interface design that meets the FCC Part 68 rules. A 2.13 meter (7 foot) modular telephone cord is included with the modem adapter.

Line-status detection of dial tone, ringback tone, busy, and incoming ring is provided along with automated routines which react to detected conditions.

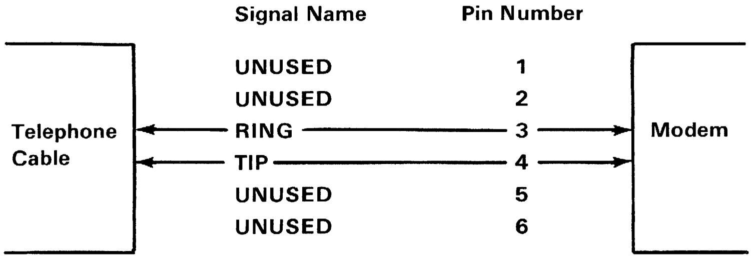
The modem card has one USOC RJ11 jack.

System I/O Channel

The following shows pin assignments for the system board modem connector. Pins A1 to A15 are on the component side.



Internal Modem Connectors



Signal Name	Pin Number
UNUSED	1
UNUSED	2
RING	3
TIP	4
UNUSED	5
UNUSED	6

Connector Specifications (Part 1 of 2)

Signal Name		Internal Modem			Signal Name
+5 Vdc	←	B01	A01	←	D7
-MODEM CS/DISKETTE CS	←			←	D6
A9	←			←	D5
-RESET	←			—	GND
GND	—	B05	A05	←	D4
A2	←			←	D3
A1	←			←	D2
A0	←			—	GND
GND	—			←	D1
-IOR	←	B10	A10	←	D0
-IOW	←			→	MODEM INTERRUPT
GND	—			—	GND
-CARD INSTALL	→			←	BAUDCLK
+XRESET	←			←	+5 Vdc
+5 Vdc	←	B15	A15	←	+12 Vdc

All levels are LSTLL compatible.

Connector Specifications (Part 2 of 2)

IBM PC*jr* Attachable Joystick

The Attachable Joystick is an input device intended to provide the user with two-dimensional positioning-control. Two pushbutton switches on the joystick give the user additional input capability.

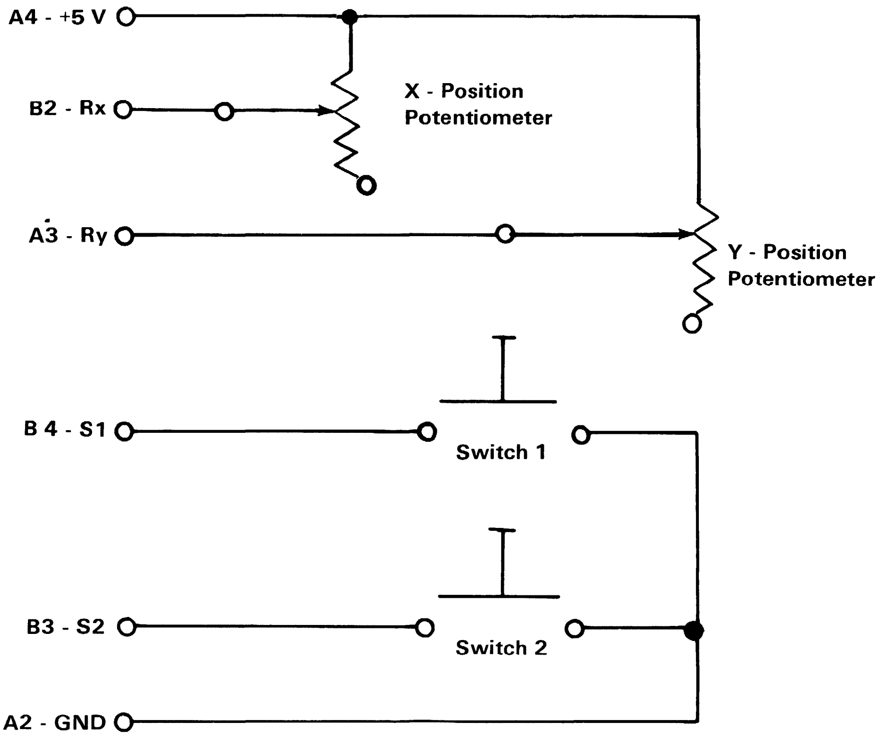
Hardware Description

Two modes of operation of the joystick are available. In the "Spring Return" mode the control stick returns to the center position when released. The "Free Floating" mode allows smooth, force free operation with the control stick remaining in position when released. Selection of these modes can be made for each axis independently. Two controls are provided for individual adjustment to the electrical center of each axis.

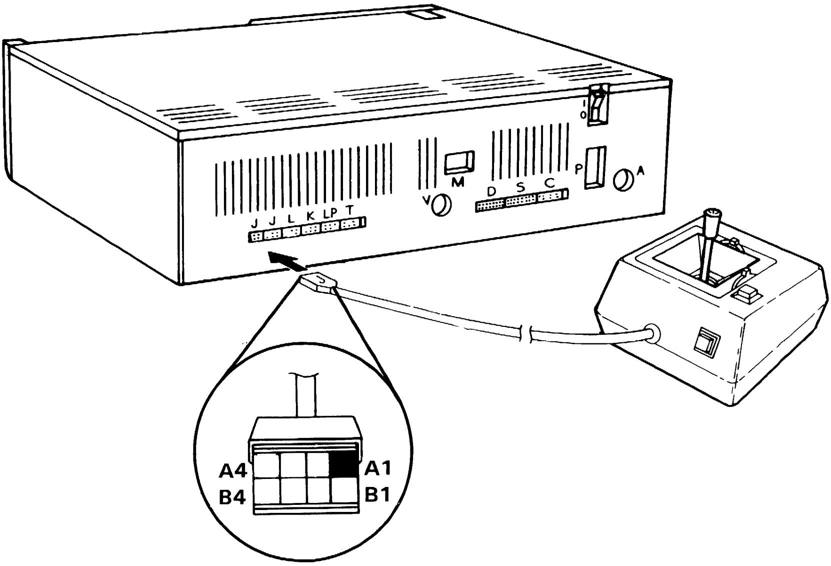
Functional Description

Positional information is derived from two potentiometers Rx and Ry. The resistance of these potentiometers will vary from 0 to 100K ohms nominally as the position of the control stick moves from left to right (X-axis) and from top to bottom (Y-axis). A linear taper is used on the potentiometers so that a linear relationship exists between angular displacement of the stick and the resulting resistance. Electrical centering for each axis is accomplished with the controls by mechanically rotating the body of the potentiometer. Adjustment in this manner has the effect of varying the minimum and maximum resistance relative to the extremes of the angular displacement. The two pushbuttons provided on the joystick are single-pole, single-throw, normally-open pushbuttons.

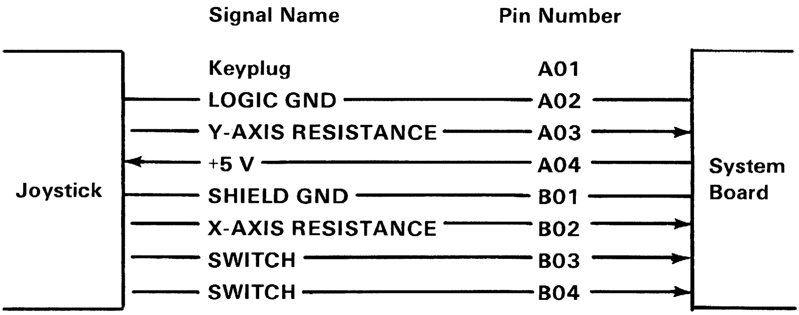
The following are the logic diagram and specifications for the two Attachable Joystick connectors.



Attachable Joystick Logic Diagram



Attachable Joystick Connector



Joystick	Signal Name	Pin Number	System Board
	Keyplug	A01	
	LOGIC GND	A02	
	Y-AXIS RESISTANCE	A03	
	+5 V	A04	
	SHIELD GND	B01	
	X-AXIS RESISTANCE	B02	
	SWITCH	B03	
	SWITCH	B04	

Connector Specifications

Notes:

IBM Color Display

The IBM Color Display is a Red/Green/Blue/Intensity (RGBI)-Direct-Drive display, that is independently housed and powered.

Hardware Description

The IBM Color Display's signal cable is approximately 1.5 meters (5 feet) in length. This signal cable must be attached to the IBM PC*jr* with the IBM PC*jr* Adapter Cable for the IBM Color Display which provides a direct-drive connection from the IBM PC*jr*

A second cable provides ac power to the display from a standard wall outlet. The display has its own power control and indicator. The display will accept either 120-volt 60-Hz power or 220-volt 50-Hz power. The power supply in the display automatically switches to match the applied power.

The display has a 340 mm (13 in.) CRT. The CRT and analog circuits are packaged in an enclosure so the display may be placed separately from the system unit. Front panel controls and indicators include: Power-On control, Power-On indicator, Brightness and Contrast controls. Two additional rear-panel controls are the Vertical Hold and Vertical-Size controls.

Operating Characteristics

Screen

- High contrast (black) screen.
- Displays up to 16 colors.
- Characters defined in an 8-high by 8-wide matrix.

Video Signal

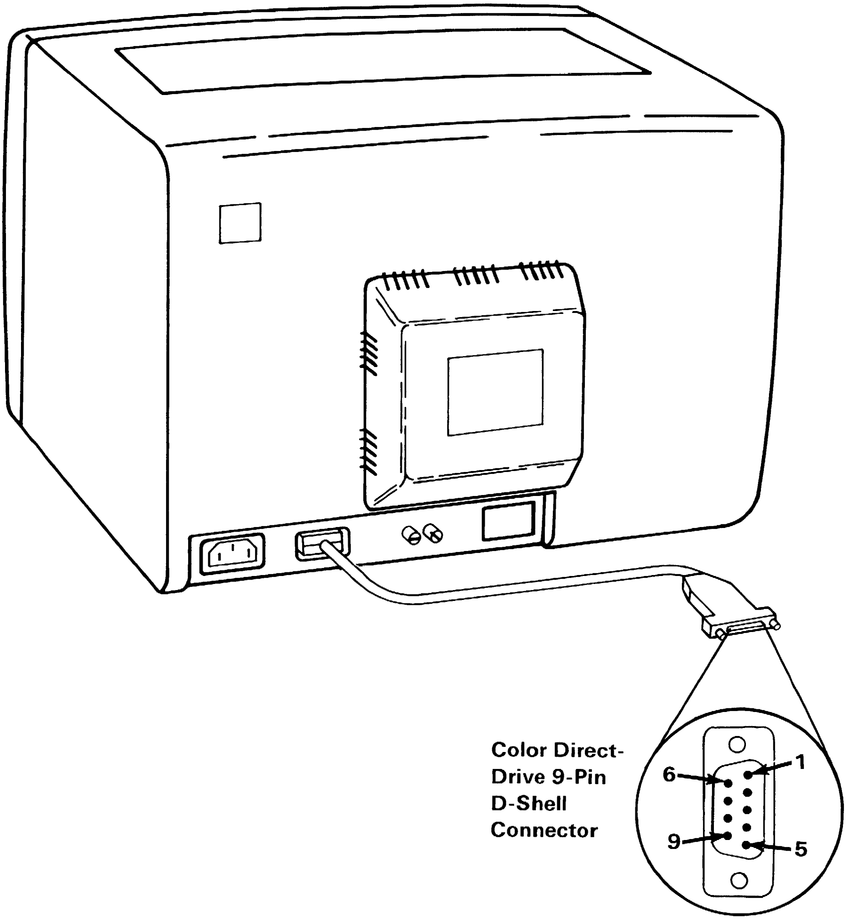
- Maximum video bandwidth of 14 MHz.
- Red, green, and blue video-signals, vertical sync, horizontial sync, and intensity are all independent. All input signals are TTL compatible.

Vertical Drive

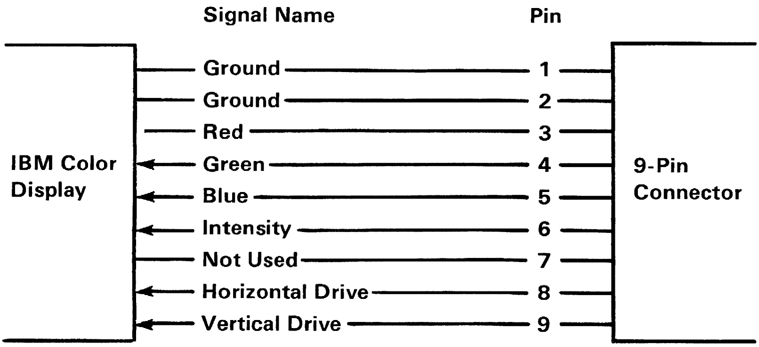
- Screen refreshed at 60 Hz with 200 vertical lines of resolution.

Horizontal Drive

- The horizontal drive frequency is 15.75 kHz.



Color-Display Connector



Signal Name	Pin
Ground	1
Ground	2
Red	3
Green	4
Blue	5
Intensity	6
Not Used	7
Horizontal Drive	8
Vertical Drive	9

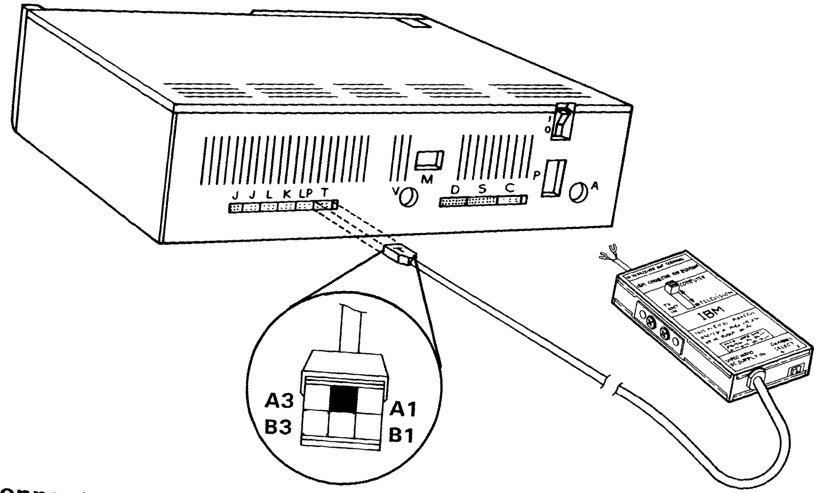
Connector Specifications

Notes:

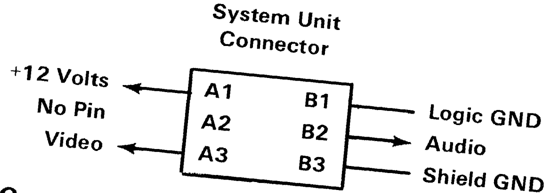
IBM Connector for Television

The Connector for Television is a sealed Radio Frequency (RF) Modulator that imposes the composite video and audio signals onto the RF carrier-wave supplied by the modulator. The connector unit has two two-position switches. One switch selects between the computer's signal or the standard-TV signal from an antenna as the input to the TV. The other switch selects either channel 3's or channel 4's carrier-wave frequency for input to the TV. This allows users to select the weaker TV channel for their area reducing the amount of interference with the computer's input signal. Signal input from the computer is provided by a five-conductor cable with a six-pin IBM PC*jr*-dedicated connector. Two spade-lug terminals provide for TV-antenna-cable connection. One twin-lead flat-type TV-cable provides input to the TV.

The following is the connector specifications for the IBM Connector for Television.



Connector for TV Connector



Connector Specifications

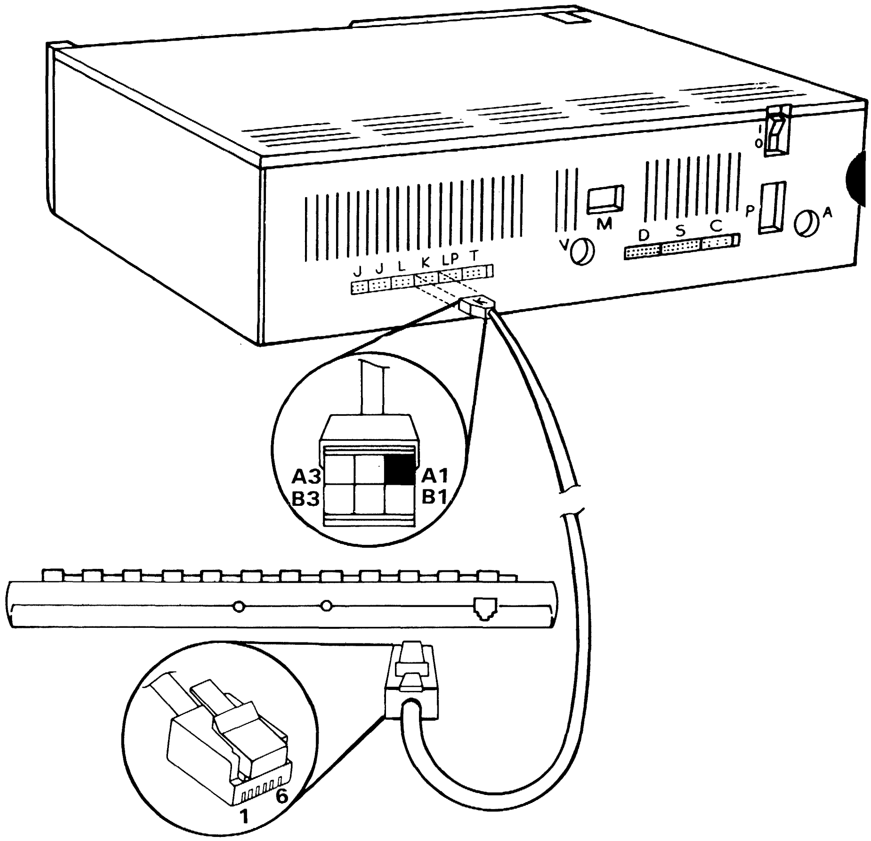
IBM PC*jr* Keyboard Cord

The IBM PC*jr* Cordless Keyboard can be attached to the PC*jr* using the optional Keyboard Cord. The Keyboard Cord is a 1.8 meter (6 foot), two twisted-pair cable, with a six-position RJ11-type connector for the keyboard and a six-position Berg-type connector for the system unit.

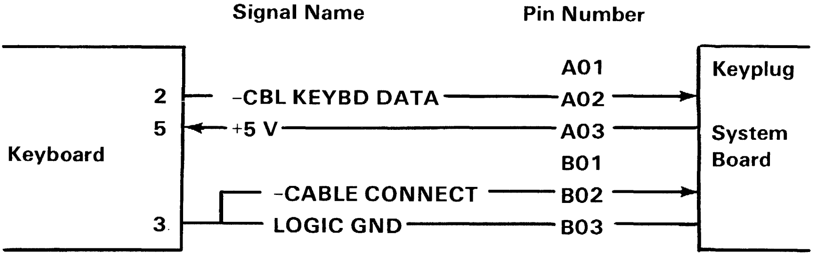
The Keyboard Cord option should be used in an environment that is unfavorable for use of the infra-red link. For instance, brightly lit high-intensity light areas, or multiple IBM PC*jr* areas where keyboards can conflict with one another.

Insertion of the cord's keyboard connector into the keyboard actuates switches internal to the keyboard. The switches 'deactivate' the IR transmitter by removing the power supplied by the keyboard's batteries. The system unit's infra-red (IR) receiver circuit is 'disabled' by the -CABLE CONNECT signal, supplied when the system-unit end of the cord is connected.

The following figures show the connector specifications for the Keyboard Cord.



Keyboard Cord Connectors

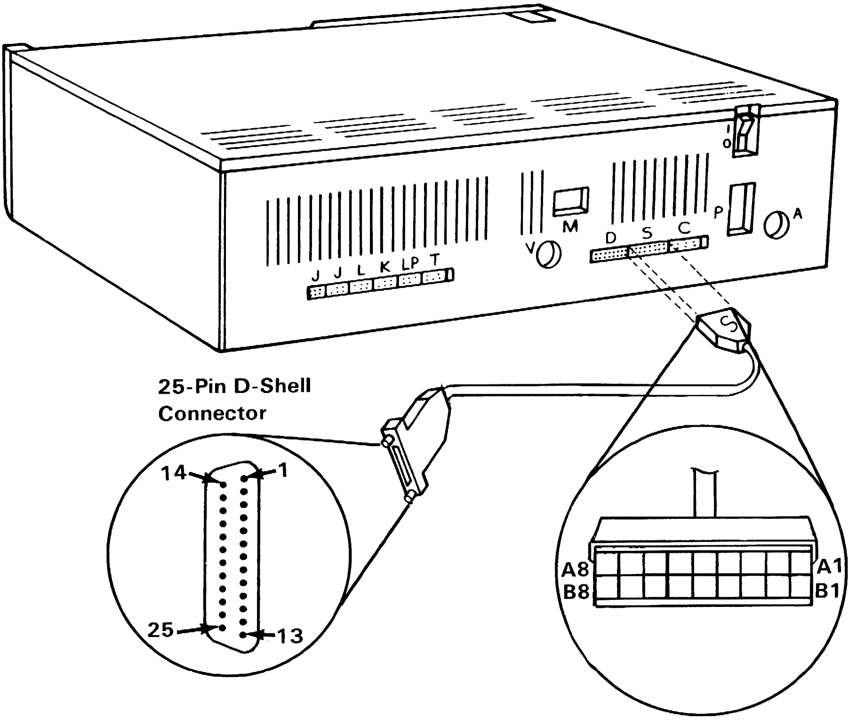


Connector Specifications

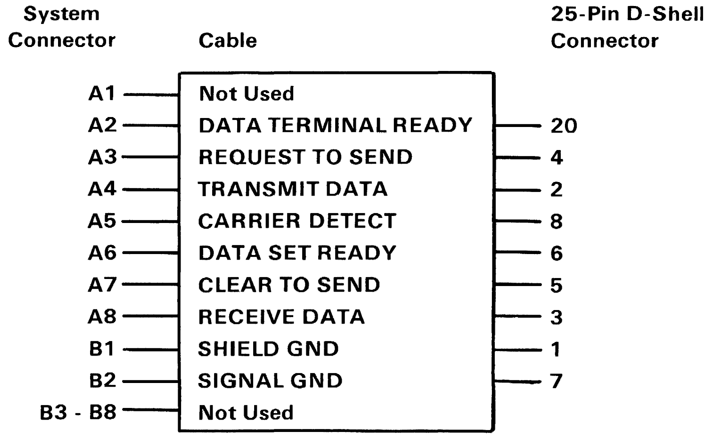
IBM PC*jr* Adapter Cable for Serial Devices

The Adapter Cable for Serial Devices is a 72 mm (3-inch) long, nine-conductor cable terminated with a 16-position Berg-type connector and a 25-pin "D"-shell connector. This cable allows serial devices that terminate with a standard EIA-RS232C 25-pin "D"-shell connector to be connected to the IBM PC*jr*.

The following figures show the connector specifications for the Adapter Cable for Serial Devices.



Adapter Cable for Serial Devices



System Connector	Cable	25-Pin D-Shell Connector
A1	Not Used	
A2	DATA TERMINAL READY	20
A3	REQUEST TO SEND	4
A4	TRANSMIT DATA	2
A5	CARRIER DETECT	8
A6	DATA SET READY	6
A7	CLEAR TO SEND	5
A8	RECEIVE DATA	3
B1	SHIELD GND	1
B2	SIGNAL GND	7
B3 - B8	Not Used	

Connector Specifications

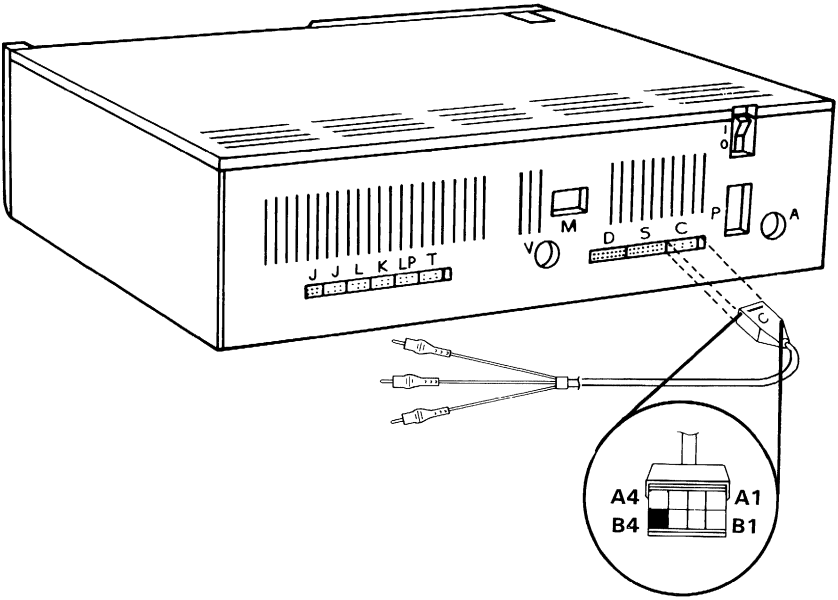
IBM PC*jr* Adapter Cable for Cassette

This option is an adapter cable that allows connection of a cassette recorder to the IBM PC*jr* cassette connector.

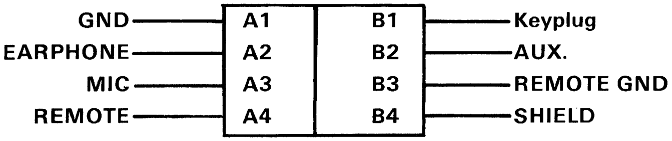
The cassette recorder to be connected must use the following type connectors:

- Belden Style-51 miniture phone-plug (Auxiliary)
- Belden Style-51 miniture phone-plug (Earphone)
- Belden Style-56 subminiture phone-plug (Remote)

The following figures show the connector specifications for the Adapter Cable for Cassette.



Adapter Cable for Cassette Connectors



Connector Specifications (System End)
(Part 1 of 2)

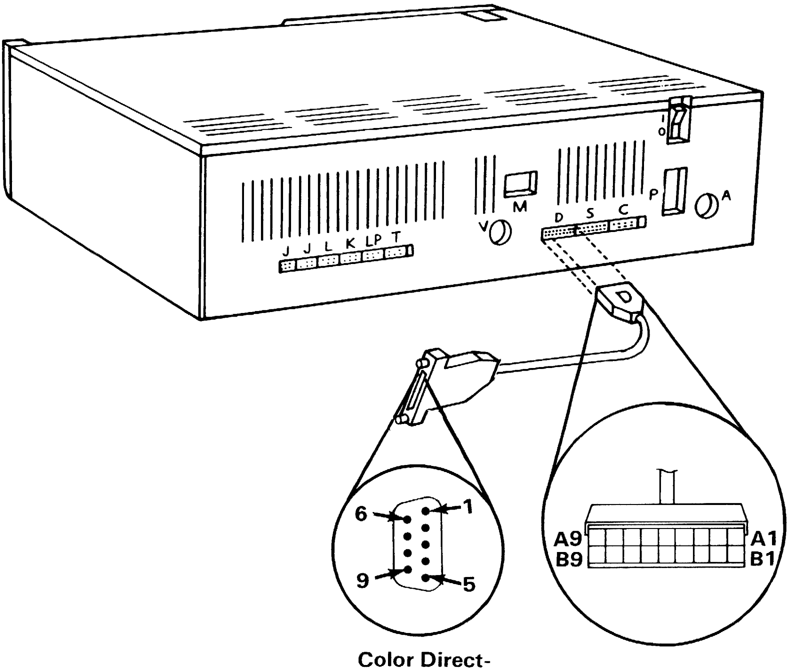
Cassette Connector		System Connector Pin
Aux. (Red)	Signal	B2
	Gnd	A1
Ear (Black)	Signal	A2
	Gnd	A1
Remote (Gray)	Signal	A4
	Gnd	B3

Connector Specifications (Recorder End)
(Part 2 of 2)

IBM PC*jr* Adapter Cable for the IBM Color Display

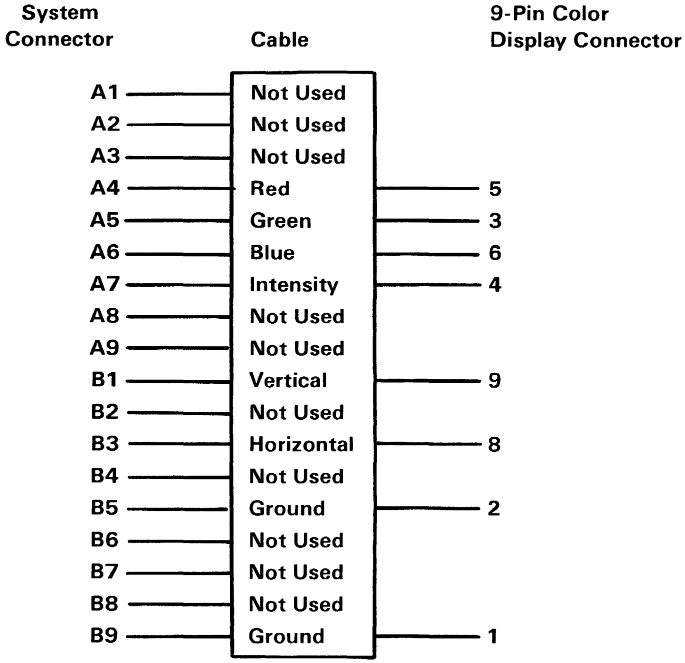
This adapter cable allows the IBM Color Display to be connected to the IBM PC*jr*.

The following figures show the connector specifications for the adapter cable for the IBM Color Display.



Color Direct-Drive 9-Pin D-Shell Connector

Adapter Cable for IBM Color Display Connectors



System Connector	Cable	9-Pin Color Display Connector
A1	Not Used	
A2	Not Used	
A3	Not Used	
A4	Red	5
A5	Green	3
A6	Blue	6
A7	Intensity	4
A8	Not Used	
A9	Not Used	
B1	Vertical	9
B2	Not Used	
B3	Horizontal	8
B4	Not Used	
B5	Ground	2
B6	Not Used	
B7	Not Used	
B8	Not Used	
B9	Ground	1

Connector Specifications

IBM PC*jr* Parallel Printer Attachment

The Parallel Printer Attachment is provided to attach various I/O devices that accept eight bits of parallel data at standard TTL-logic levels. The card measures 76mm (3 inches) high by 244mm (9.6 inches) long.

The Parallel Printer Attachment attaches as a feature to the right-hand side of the system unit. It connects to the 60-pin Input/Output (I/O) connector where power and system-input signals are received. A parallel printer attaches to the Parallel Printer Attachment through a 25-pin female "D"-shell connector located on the rear edge of the attachment, where a cable and shield can be attached. The logic design is compatible with the IBM Personal Computer printer adapter.

The attachment card has 12 TTL buffer-output points which are latched and can be 'written' and 'read' under program control using the processor 'IN' or 'Out' instructions. The attachment card also has five steady-state input-points that may be 'read' using the processors' 'IN' instructions.

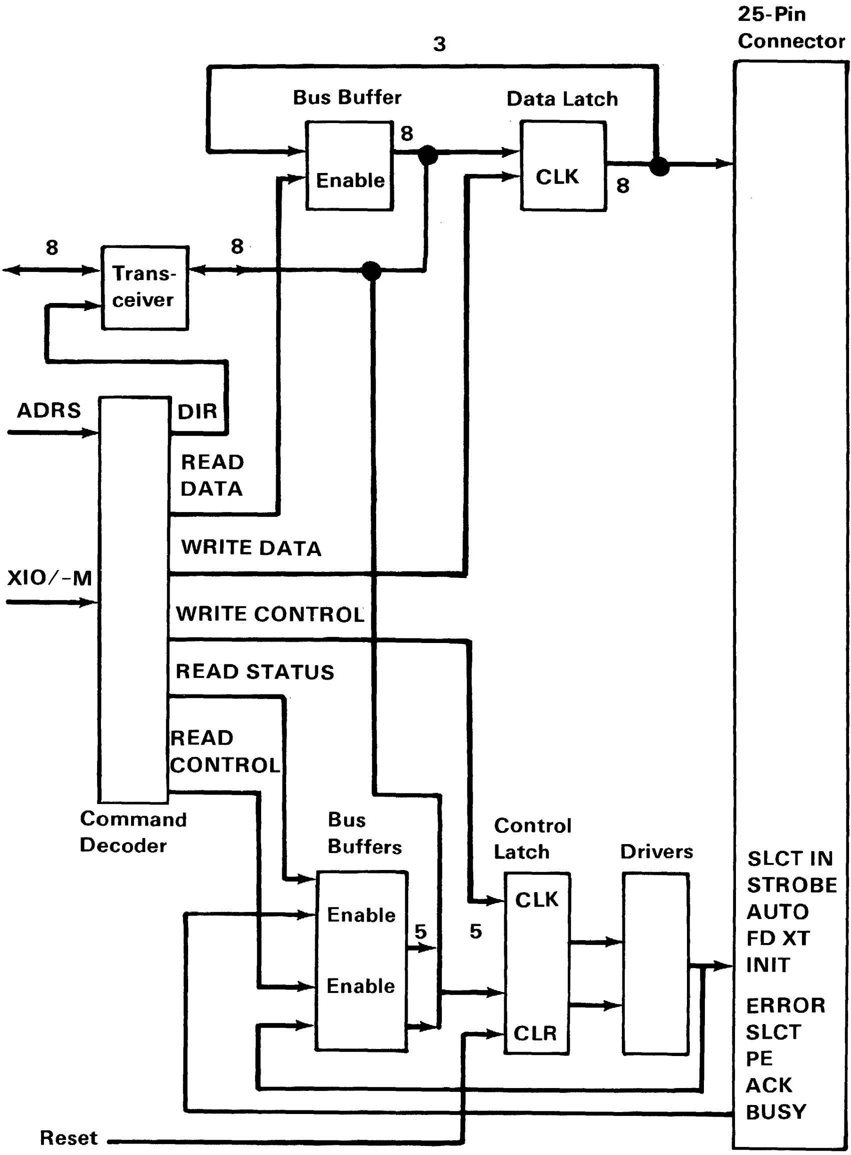
In addition, one input can also be used to create a processor interrupt. This interrupt can be 'enabled' and 'disabled' under program control. 'Reset' from the power-on circuit is also **ORed** with a program-output point allowing a device to receive a power-on 'reset' when the processor is 'reset.'

When the Parallel Printer Attachment is used to attach a printer, data or printer commands are loaded into an 8-bit latched output-port, then the strobe line is 'activated' to 'write' data to the printer. The program can then 'read' the input ports for printer status indicating when the next character can be written or it may use the interrupt line to indicate **not busy** to the software.

The output ports can also be 'read' at the card's interface for diagnostic-loop functions. This allows fault-isolation determination between the printer attachment and the attached printer.

Description

During a system I/O 'read' or 'write', with the proper address selection, data may be 'written' to or 'read' from the Parallel Printer Attachment. The data and Control Registers must be manipulated by the system software to be consistent with the attaching hardware. The following is a block diagram of the Parallel Printer Attachment card.



Parallel Printer Interface Block Diagram

System Interface

The Parallel Printer Attachment reserves addresses hex 378, through hex 37F. **IO/-M** must also be 'active high' when addressing the Parallel Printer Attachment.

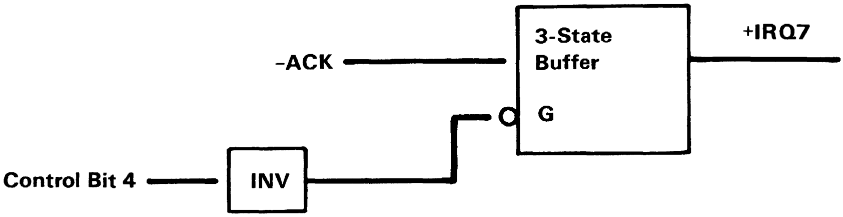
A card selected signal **(-CARD SLCTD)** is provided to the system I/O when the above addresses are used, and the **IO/-M** bit is 'active high.'

Specific commands are decoded from A0, A1, RD, and WR per the following table. Input A2 is not used.

Addresses (hex)	**Operation**	**Comments**
378	'Read'	Read Data Latch
379	'Read'	Read Status
37A	'Read'	Read Control Latch
37B	'Read'	Unused
37B	'Write'	Write Data Latch
379	'Write'	Unused
37A	'Write'	Write Control Latch
37B	'Write'	Unused

All data transfers take place over the 8-bit I/O data-bus with timing provided by the 8088 microprocessor. (IOR, IOW, IO/-M)

An interrupt is provided to the system through the I/O connector of the Parallel Printer Attachment. This interrupt is 'positive active', Interrupt Level 7 (+IRQ7). Bit 4 of the control latch must be 'written high' to allow interrupts. When the -ACKnowledge signal ('low active' signal goes 'high') the I/O device causes a level 7 interrupt. See the following figure.



+IQR7/-ACK Logic Diagram

Programming Considerations

The Parallel Printer Attachment can serve as a general purpose peripherial driver. This section describes a configuration which supports attachment to the IBM Graphics Printer.

Command Definition

For the parallel-printer application, the following bit definitions apply.

Data Latch - Address hex 378

A 'write' to this address causes data to be latched onto the printer data bits. A 'read' from this address presents the contents of the data latch to the processor.

MSB	7	6	5	4	3	2	1	0	LSB
	Data	Data	Data	Data	Data	Data	Data	Data	
	Bit	Bit	Bit	Bit	Bit	Bit	Bit	Bit	
	7	6	5	4	3	2	1	0	

Data Latch Format

Printer Status - Address hex 379, hex 7D, Input Only

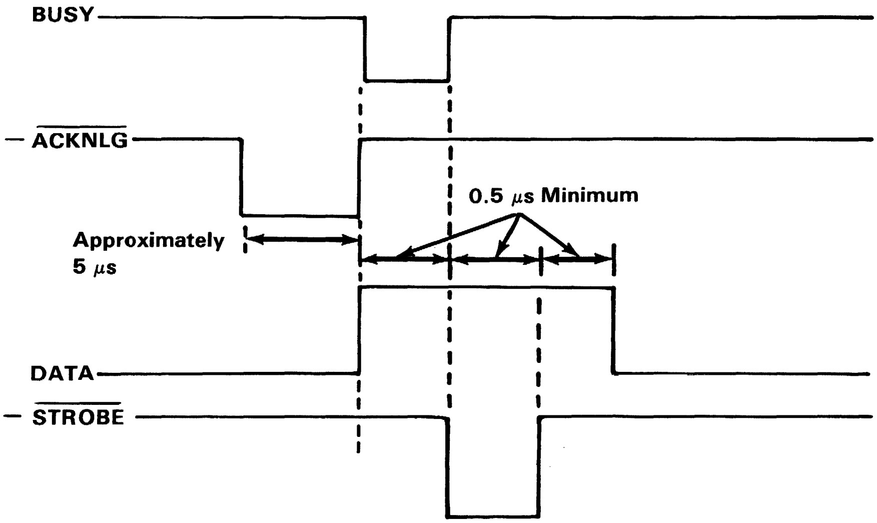
This port provides real-time feedback and status to the system from the printer.

Bit	Signal Name	Description
MSB 7	-BUSY	When this signal is at a low level, the printer is busy and cannot accept data. It can become low during data entry, off-line printing, head translation, or error state.
6	-ACK	When port B is read, this bit will represent the current state of the printer ACK signal. A low level means that a character has been received and the printer is ready to accept another. Normally, this signal will be low for approximately 5 microseconds before BUSY goes away.
5	-PE	A low level indicates that the printer has detected an end of form.
4	+SLCT	A high level indicates that the printer is selected.
3	-ERROR	A low level indicates that the printer has encounted an error condition.
2 Through 0 LSB		Unused.

Printer Status

Printer Control - Address hex 37A

This port contains printer control signals. A 'write' latches control bits to the printer; a 'read' presents the contents of the latches to the processor. See the following timing diagram:



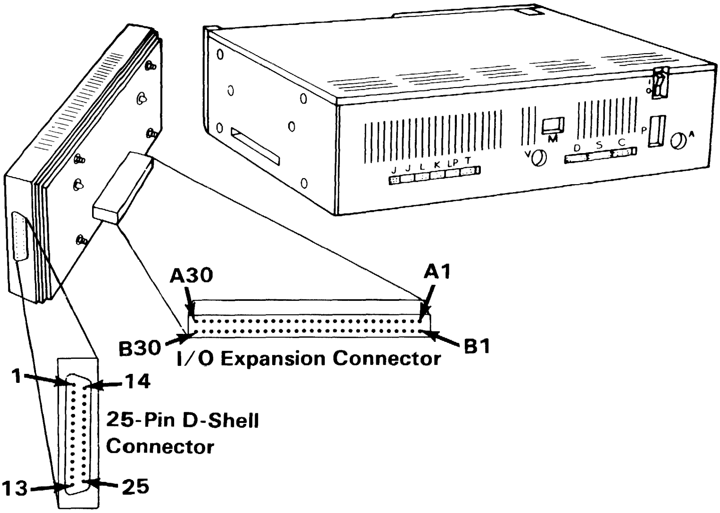
Parallel Interface Timing Diagram

The following figure describes the printer control signals.

Bit	Signal Name	Description
MSB 7 Through 5		Unused.
4	+INTERRUPT ENABLE	A high level in this bit position will allow an interrupt to occur when -ACK goes high.
3	SLCT IN	A low level in this bit position selects the printer.
2	INIT	A low level will initialize the printer (50 microseconds minimum).
1	AUTO FD XT	A low level will cause the printer to line feed anytime a line is printed.
LSB 0	STROBE	A 5 microsecond (minimum) low active pulse clocks data into the printer. Valid data must be present for 5 microseconds (minimum) before and after the STROBE pulse.

Printer Control Signal

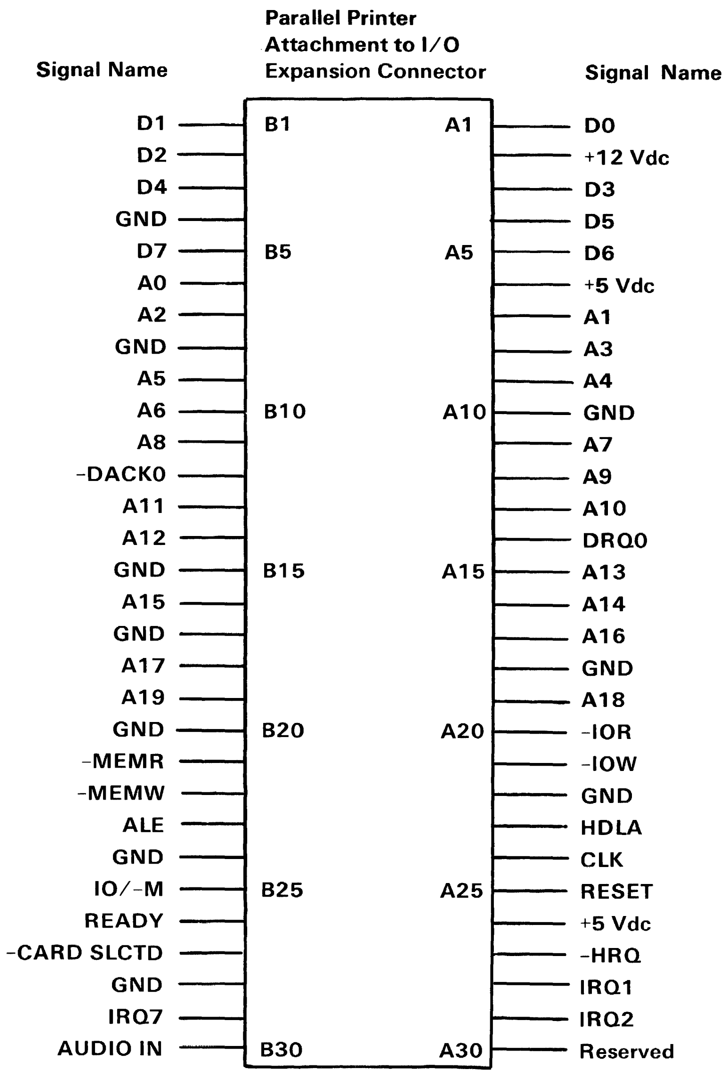
The following are the connector specifications for the IBM PC*jr* Parallel Printer Attachment.



Parallel Printer Attachment Connectors

25-Pin "D"-Shell Connector				
Pin	**Signal**	**I_{OL} Max**	**I_{OH} Max**	**Source**
1	-STROBE	14 ma	-.6 ma	Attachment Card
2 Through 9	DATA BIT 0 Through DATA BIT 7	24 ma	-2.6 ma	Attachment Card
10	-ACK	74LS Input	74LS Input	Printer
11	BUSY	74LS Input	74LS Input	Printer
12	PE	74LS Input	74LS Input	Printer
13	SLCT	74LS Input	74LS Input	Printer
14	-AUTO FD XT	14 ma	.6 ma	Attachment Card
15	-ERROR	74LS Input	74LS Input	Printer
16	-INIT PRINTER	14 ma	.6 ma	Printer
17	-SELECT INPUT	14 ma	.6 ma	Attachment Card
18 Through 25	GND	N/A	N/A	

Connector Specifications (Part 1 of 2)



Signal Name	Parallel Printer Attachment to I/O Expansion Connector		Signal Name
D1	B1	A1	D0
D2			+12 Vdc
D4			D3
GND			D5
D7	B5	A5	D6
A0			+5 Vdc
A2			A1
GND			A3
A5			A4
A6	B10	A10	GND
A8			A7
-DACK0			A9
A11			A10
A12			DRQ0
GND	B15	A15	A13
A15			A14
GND			A16
A17			GND
A19			A18
GND	B20	A20	-IOR
-MEMR			-IOW
-MEMW			GND
ALE			HDLA
GND			CLK
IO/-M	B25	A25	RESET
READY			+5 Vdc
-CARD SLCTD			-HRQ
GND			IRQ1
IRQ7			IRQ2
AUDIO IN	B30	A30	Reserved

Connector Specifications (Part 2 of 2)

Notes:

IBM Graphics Printer

The IBM Graphics Printer is a self-powered, stand-alone, tabletop unit which attaches to the system unit through a 6-foot parallel-signal cable, and obtains 120 Vac power from a standard wall outlet through a seperate cable. It is an 80 CPS (characters per second), bidirectional, wire-matrix device that can print in a compressed mode of 132 characters per line, in a standard mode of 80 characters per line, in a double width-compressed mode of 66 characters per line, and in a double width mode of 40 characters per line. It can also print double-size and double-strike characters. It prints the standard ASCII, 96-character, uppercase and lowercase character sets and also has a set of 64 special block characters. It has an extended character set for international languages, subscript, superscript, an underline mode, and programmable graphics. The Graphics printer accepts commands that set the line-feed control desired for the application.

It attaches to the system unit through the IBM PC*jr* Parallel Printer Attachment. The cable is a 25-conducter, shielded cable with a 25-pin "D"-shell connector at the system unit end, and a 36-pin connector at the printer end.

Printer Specifications

Print Method: Serial-impact dot matrix

Print Speed: 80 CPS

Print Direction: Bidirectional with logic seeking

Number of Pins in Head: 9

Line Spacing: 1/16 inch (4.23 mm) or programmable

Matrix Characteristics: 9 by 9

Character Set: Full 96-character ASCII with descenders plus 9 international characters/symbols

Graphic Characters: See "Additional Printer Specifications"

Printing Sizes:

Normal	10 characters-per-inch with a maximum of 80 characters-per-line
Double Width	5 characters-per-inch with a maximum of 40 characters per line
Compressed	16.5 characters-per-inch with a maximum of 132 characters per line
Double Width-Compressed	8.25 characters-per-inch with a maximum of 66 characters per line
Subscript	10 characters-per-inch with a maximum of 80 characters per line
Superscript	10 characters-per-inch with a maximum of 80 characters per line

Media Handling: Adjustable sprocket-pin-feed with 4-inch (101.6 mm) to 10-inch (254 mm) width paper, one original plus two carbon copies (total thickness not to exceed 0.012 inch (0.3 mm)), minimum paper thickness of 0.0025 inch (0.064 mm)

Interface: Parallel 8-bit data and control lines

Inked Ribbon: Black, cartridge type with a life expectancy of 3 million characters

Environmental Conditions: Operating temperature is 5 to 35 degrees centigrade (41 to 95 degrees Fahrenheit), operating humidity is 10 to 80% non-condensing

Power Requirements: 120 Vac, 60 Hz, 1 A maximum with a power consumption of 100 VA maximum

Physical Characteristics:

Height	107 mm (4.2 inches)
Width	374 mm (14.7 inches
Depth	305 mm (12 inches)
Weight	5.5 kg (12 pounds)

Additional Printer Specifications

Printing Characteristics

Extra Character Set

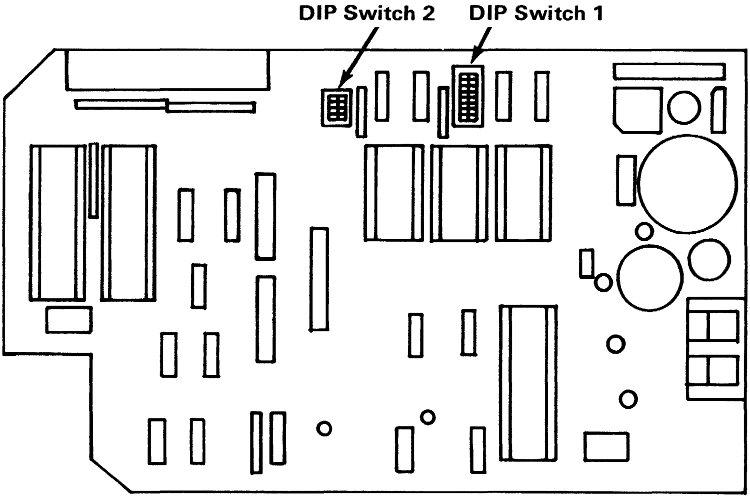
Set 1 Additional ASCII numbers 160 to 175 contain European characters. Numbers 176 to 223 contain graphic characters. Numbers 224 to 239 contain selected Greek-characters. Numbers 240 to 255 contain math and extra symbols.

Set 2 The differences in Set 2 are ASCII numbers 3,4,5,6, and 21. ASCII numbers 128 to 175 contain European characters.

Graphics There are 20 block characters and programmable graphics.

DIP Switch Settings

There are two Dual-Inline-Package (DIP) switches on the control circuit-board. In order to satisfy the user's specific requirements, desired control modes are selected by the DIP switches. The functions of these switches and their preset conditions at the time of shipment are shown in the following figures.



Location of DIP Switches

Switch Number	Function	On	Off	Factory Position
1-1	Not Applicable	—	—	On
1-2	CR	Print Only	Print and Line Feed	On
1-3	Buffer Full	Print Only	Print and Line Feed	Off
1-4	Cancel Code	Invalid	Valid	Off
1-5	Not Applicable	—	—	On
1-6	Error Buzzer	Sound	No Sound	On
1-7	Character Generator	Set 2	Set 1	Off
1-8	SLCT IN Signal	Fixed Internally	Not Fixed Internally	On

Functions and Conditions of DIP Switch 1

Switch Number	Function	On	Off	Factory Position
2-1	Form Length	12 Inches	11 Inches	Off
2-2	Line Spacing	1/8 Inch	1/6 Inch	Off
2-3	Auto Feed XT Signal	Fixed Internally	Not Fixed Internally	Off
2-4	1 Inch Skip Over Perforation	Valid	Invalid	Off

Functions and Conditions of DIP Switch 2

Parallel Interface Description

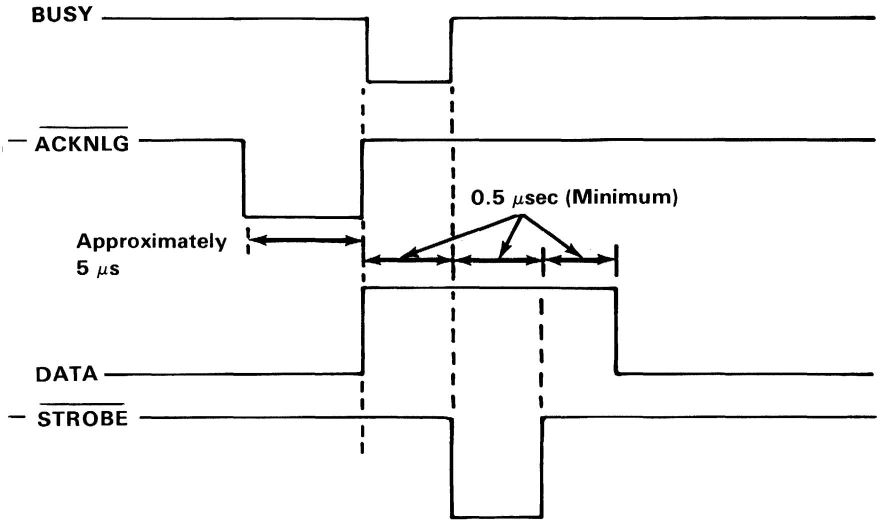
Specifications

Data Transfer Rate	1000 cycles-per-second (cps)-(maximum)
Synchronization	By externally-supplied STROBE pulses
Signal Exchange	-ACKNLG or BUSY signals
Logic level	Input data and all interface-control signals are compatible with the Transistor-Transistor Logic (TTL) level.
Connector	Plug 57-30360 (Amphenol)

Connector-pin assignments and descriptions of respective interface-signals are provided in the following figures.

Data Transfer Sequence

The following figure shows the Parallel Interface Timing.



Parallel Interface Timing Diagram

Interface Signals

-Strobe STROBE pulse to read data in. Pulse width must be more than 0.5 μs at the receiving terminal. The signal is normally 'high'; however read-in of data is performed at the 'Low' level of this signal.

Data 1-8 These signals are the first to eight bits of parallel data. Each signal is at a 'high' level when data is a logical 1 and 'low' when data is a logical 0.

-ACKNLG Approximately 0.5 μs pulse (low) indicates that data has been received and the printer is ready to accept data.

BUSY A 'high' signal indicates that the printer cannot receive data. The signal is 'high' in the following cases:

- During data entry

- During printing operation
- In the “off-line” state
- During printer-error status

PE A 'high' signal indicates that the printer is out of paper.

SLCT This signal indicates that the printer is in the selected state.

Auto Feed XT When this signal is 'low' paper is fed one line after printing. This signal level can be fixed 'low' by DIP switch pin 2-3.

INT When this signal is 'low' the printer controller is reset to its initial state and the print buffer is cleared. This signal is normally 'high' and its pulse width must be more than 50 μs at the receiving terminal.

Error This signal is 'low' when the printer is in the “Paper End,” “Off Line,” and “Error” state.

-SLCTIN Data entry to the printer is possible only when this signal is 'low'. This signal can be fixed 'low' by DIP switch 1-8.

Notes:

1. All interface conditions are based on TTL level. Both the rise and fall times of each signal must be less than 0.2 μs.

2. Data transfer must not be carried out by ignoring the -ACKNLG or BUSY signal. Data transfer can only occur after confirming the -ACKNLG signal or when the BUSY signal is 'low'.

The following figure shows the pin assignment and direction of each signal.

Signal	Signal Pin #	Return Pin #	Direction
-STROBE	1	19	In
DATA 1	2	20	In
DATA 2	3	21	In
DATA 3	4	22	In
DATA 4	5	23	In
DATA 5	6	24	In
DATA 6	7	25	In
DATA 7	8	26	In
DATA 8	9	27	In
-ACKNLG	10	28	Out
BUSY	11	29	Out
PE	12	30	Out
SLCT	13	—	Out
AUTO FEED XT	14	—	In
NC	15	—	—
OV	16	—	—
CHASSIS GND	17	—	—
NC	18	—	—
GND	19-30	—	—
INT	31	—	In
ERROR	32	—	Out
GND	33	—	—
NC	34	—	—
	35	—	—
-SLCT IN	36	—	In

Pin Assignments

Printer Modes

The IBM Graphics Printer can use any of the combinations listed in the following table and the print mode can be changed at any place within the line.

Modes can be selected and combined if they are in the same vertical column.

Printer Modes									
Normal	X	X	X						
Compressed				X	X	X			
Emphasized							X	X	X
Double Strike	X			X			X		
Subscript		X			X			X	
Superscript			X			X			X
Double Width	X	X	X	X	X	X	X	X	X
Underline	X	X	X	X	X	X	X	X	X

Printer Modes

Printer Control Codes

On the following pages are complete codes for printer characters, controls, and graphics. You may want to keep them handy for future reference. The printer codes are listed in ASCII-decimal numeric-order (from NUL which is 0 to DEL, which is 127). The examples given in the Printer-Function descriptions are written in the BASIC language. The "input" description is given when more information is needed for programming considerations.

ASCII decimal values for the printer control codes can be found under "Printer Character Sets."

The Descriptions that follow assume that the printer DIP switches have not been changed from their factory settings.

Printer code — **Printer Function**

NUL — Null:

Used with ESC B and ESC D as a list terminator. NUL is also used with other printer.

control codes to select options (for example, ESC S).
Example:

```
LPRINT CHR$ (0);
```

BEL Bell:
Sounds the printer buzzer for 1 second.
Example:

```
LPRINT CHR$(7);
```

HT Horizontal Tab:
Tabs to the next horizontal tab stop. Tab stops are set with ESC D. Tab stops are set every 8 columns when the printer is powered on.
Example:

```
LPRINT CHR$(9);
```

LF Line Feed:
Spaces the paper up one line. Line spacing is 1/16-inch unless reset by ESC A, ESC 0, ESC 1, ESC 2, or ESC 3.
Example:

```
LPRINT CHR$(10);
```

FF Form Feed:
Advances the paper to the top of the next page.

> **Note:** The location of the paper, when the printer is powered on, determines the top of the page. The next top of page is 11 inches from that position. ESC C can be used to change the page length.

Example:

```
LPRINT CHR$(12);
```

CR Carriage Return:
Ends the line that the printer is on and prints the data remaining in the printer buffer. (No Line Feed operation takes place.)

Note: IBM Personal Computer BASIC adds a Line Feed unless 128 is added [for example CHR$(141)].

Example:
LPRINT CHR$(13);

SO Shift Out (Double Width):
Changes the printer to the Double-Width print-mode.

Note: A Carriage Return, Line Feed or DC4 cancels Double-Width print-mode.

Example:
LPRINT CHR$(14);

SI Shift In (Compressed):
Changes the printer to the Compressed-Character print-mode. Example:
LPRINT CHR$(15);

DC2 Device Control 1 (Compressed Off):
Stops printing in the Compressed print-mode.
Example:
LPRINT CHR$(18);

DC4 Device Control 4 (Double Width Off):
Stops printing in the Double-Width print-mode.
Example:
LPRINT CHR$(20);

CAN Cancel:
Clears the printer buffer. Control codes, except SO, remain in effect.
Example:
LPRINT CHR$(24);

ESC Escape:
Lets the printer know that the next data sent is a printer command.
Example:
LPRINT CHR$(27);

ESC - Escape Minus (Underline)

Format: ESC -;n;

ESC - followed by a 1, prints all of the following data with an underline.

ESC - followed by a 0 (zero), cancels the Underline print-mode.

Example:

LPRINT CHR$(27);CHR$(45);CHR$(1);

ESC 0 Escape Zero (1/8-Inch Line Feeding)

Changes paper feeding to 1/8-inch.

Example:

LPRINT CHR$(27);CHR$(48);

ESC 1 Escape One (7/72-Inch Line Feeding)

Changes paper feeding to 7/72-inch.

Example:

LPRINT CHR$(27);CHR$(49);

ESC 2 Escape Two (Starts Variable Line-Feeding)

ESC 2 is an execution command for ESC A. If no ESC A command has been given, line feeding returns to 1/6-inch.

Example:

LPRINT CHR$(27);CHR$(50);

ESC 3 Escape Three (Variable Line-Feeding)

Format: ESC 3;n;

Changes the paper feeding to n/216-inch. The example that follows sets the paper feeding to 54/216 (1/4)-inch. The value of n must be between 1 and 255.

Example:

LPRINT CHR$(27);CHR$(51);CHR$(54);

ESC 6 Escape Six (Select Character Set 2)

Selects Character Set 2. (See "Printer Character set 2")

Example:

LPRINT CHR$(27);CHR$(54);

ESC 7 Escape Seven (Select Character Set 1)

Selects character set 1. (See "Printer Character Set 1")
Character set 1 is selected when the printer is powered on or reset.
Example:
LPRINT CHR$(27);CHR$(55);

ESC 8 Escape Eight (Ignore Paper End)
Allows the printer to print to the end of the paper. The printer ignores the Paper End switch.
Example:
LPRINT CHR$(27);CHR$(56);

ESC 9 Escape Nine (Cancel Ignore Paper End)
Cancels the Ignore Paper End command. ESC 9 is selected when the printer is powered on or reset.
Example:
LPRINT CHR$(27);CHR$(57);

ESC < Escape Less Than (Home Head)
The printer head returns to the left margin to print the line following ESC <. This occurs for one line only.
Example:
LPRINT CHR$(27);CHR$(60);

ESC A Escape A (Sets Variable Line Feeding)
Format: ESC A;n;
Escape A sets the line-feed to n/72-inch. The example that follows tells the printer to set line feeding to 24/72-inch. ESC 2 must be sent to the printer before the line feeding changes. For example, ESC A;24 (text) ESC 2 (text). The text following ESC A;24 spaces at the previously set line-feed increments. The text following ESC 2 prints with new line-feed increments of 24/72-inch. Any increment between 1/72 and 85/72-inch may be used.

Example:

```
LPRINT
CHR$(27);CHR$(65);CHR$(24);
CHR$(27);CHR$(50);
```

ESC C

Escape C (Set Lines-per-Page)
Format: ESC C;n;
Sets the page length. The ESC C command must have a value following it to specify the length of page desired. (Maximum form length for the printer is 127 lines.) The example below sets the page length to 55 lines. The printer defaults to 66 lines-per-page when powered on or reset.
Example:

```
LPRINT CHR$(27);CHR$(67);CHR$(55);
```

Escape C (Set Inches-per-Page)
Format: ESC C;n;m;
Escape C sets the length of the page in inches. This command requires a value of 0 (zero) for n, and a value between 1 and 22 for m.
Example:

```
LPRINT CHR$(27);CHR$(67);CHR$(0);CHR$(12);
```

ESC D

Escape D (Sets Horizontal Tab Stops)
Format: ESC D;n1;n2;...nk;NUL;
Sets the horizontal-tab stop-positions. The example that follows shows the horizontal-tab stop-positions set at printer column positions of 10, 20, and 40. They are followed by CHR$(0), the NUL code. They must also be in ascending numeric order as shown. Tab stops can be set between 1 and 80. When in the Compressed-print mode, tab stops can be set up to 132.
The Graphics Printer can have a maximum of 28 tab stops. The HT (CHR$(9)) is used to execute a tab operation.
Example:

```
LPRINT
CHR$(27);CHR$(68);CHR$(10)
;CHR$(20);CHR$(40);
CHR$(0);
```

ESC E Escape E (Emphasized)

Changes the printer to the Emphasized-print mode. The speed of the printer is reduced to half speed during the Emphasized-print mode.

Example:

```
LPRINT CHR$(27);CHR$(69);
```

ESC F Escape F (Emphasized Off)

Stops printing in the Emphasized-print mode.

Example:

```
LPRINT CHR$(27);CHR$(70);
```

ESC G Escape G (Double Strike)

Changes the printer to the Double-Strike print-mode. The paper is spaced 1/216 of an inch before the second pass of the print head.

Example:

```
LPRINT CHR$(27);CHR$(71);
```

ESC H Escape H (Double Strike Off)

Stops printing in the Double-Strike mode.

Example:

```
LPRINT CHR$(27);CHR$(72);
```

ESC J Escape J (Sets Variable Line Feeding)

Format: ESC J;n;

When ESC J is sent to the printer, the paper feeds in increments of n/216 of an inch. The value of n must be between 1 and 255. The example that follows gives a line feed of 50/216-inch. ESC J is canceled after the line feed takes place.

Example:

```
LPRINT CHR$(27);CHR$(74);CHR$(50);
```

ESC K Escape K (480 Bit-Image Graphics Mode)

Format ESC K;n1;n2;v1;v2;...vk;

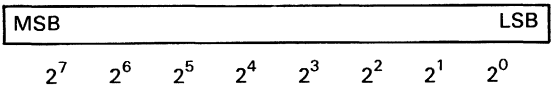
Changes from the Text mode to the Bit-Image

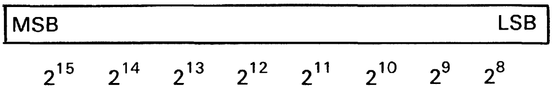
Graphics mode. n1 and n2 are one byte, which specify the number of bit-image data bytes to be transferred. v1 through vk are the bytes of the bit-image data. The number of bit-image data bytes (k) is equal to n1 +256n2 and cannot exceed 480 bytes. At every horizontal position, each byte can print up to 8 vertical dots. Bit-image data may be mixed with text data on the same line.

> **Note:** Assign values to n1 and n2 as follows:
> n1 represents values from 0 - 255.
> n2 represents values from 0 - 1 x 256.

MSB is most-significant bit and LSB is least -significant bit.

The following figures show the format.



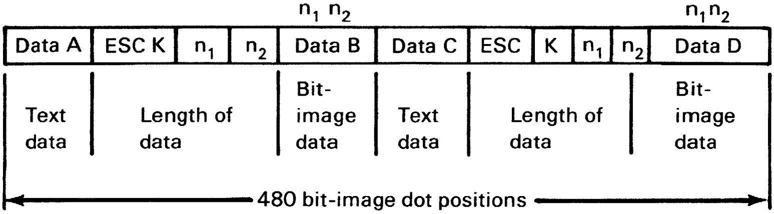


Data sent to the printer.

Text (20 characters)	ESC	K	n=360	Bit-image data	Next data

In text mode, 20 characters in text mode correspond to 120 bit-image positions (20 x 6 = 120). The printable portion left in Bit-Image mode is 360 dot positions (480 - 120 = 360).

Data sent to the printer.



```
Example: 1 'OPEN PRINTER IN RANDOM MODE
WITH LENGTH OF 255
2  OPEN "LPT1:"AS #1
3  WIDTH "LPT1:",255
4  PRINT #1,CHR$(13)+CHR$(10);
5  SLASH$=CHR$(1)+CHR$(02)
+CHR$(04)+CHR$(08)
6  SLASH$=SLASH$+CHR$(16)+CHR$(32)
+CHR&(64)+$CHR$(128)+CHR$(0)
7  GAP$=CHR$(0)+CHR$(0)+CHR$(0)
8  NDOTS=480
9  'ESC K N1 N2
10 PRINT #1,CHR$(27);"K";CHR$(NDOTS
MOD 256);CHR$ (FIX(NDOTS/256));
11 'SEND NDOTS NUMBER OF BIT
IMAGE BYTES
12 FOR I=1 TO NDOTS/12 'NUMBER
OF SLASHES TO
PRINT USING GRAPHICS
13 PRINT #1,SLASH$;GAP$;
```

```
14 NEXT I
15 CLOSE
16 END
```

This example gives you a row of slashes printed in the Bit-Image mode.

ESC L

Escape L (960-Bit-Image Graphics-Mode)
Format: ESC L;n1;n2;v1;v2;...vk;
Changes from the Text mode to the Bit-Image Graphics mode. The input is similar to ESC K. The 960 Bit-Image mode prints at half the speed of the 480 Bit-Image Graphics mode, but can produce a denser graphic image. The number of bytes of bit-image Data (k) is n1 +256n2 but cannot exceed 960. n1 is in the range of 0 to 255.

ESC N

Escape N (Set Skip Perforation)
Format ESC N;n;
Sets the Skip Perforation function. The number following ESC N sets the value for the number of lines of Skip Perforation. The example shows a 12-line skip perforation. This prints 54 lines and feeds the paper 12 lines. The value of n must be between 1 and 127. ESC N must be reset anytime the page length (ESC C) is changed.
Example:
LPRINT CHR$(27);CHR$(78);CHR$(12);

ESC O

Escape O (Cancel Skip Perforation)
Cancels the Skip Perforation function.
Example:
LPRINT CHR$(27);CHR$(79);

ESC S

Escape S (Subscript/Superscript)
Format: ESC S;n;
Changes the printer to the Subscript print mode when ESC S is followed by a 1, as in the example that follows. When ESC S is followed by a 0 (zero), the printer prints in the

Superscript print mode.
Example:
LPRINT CHR$(27);CHR$(83);CHR$(1);

ESC T Escape T (Subscript/Superscript Off)
The printer stops printing in the Subscript or Superscript print mode.
Example:
LPRINT CHR$(27);CHR$(84);

ESC U Escape U (Unidirectional Printing)
Format: ESC U;n;
The printer prints from left to right following the input of ESC U;1. When ESC U is followed by a 0 (zero), the left to right printing operation is canceled. The Unidirectional print-mode (ESC U) ensures a more accurate print-start position for better print quality.
Example:
LPRINT CHR$(27);CHR$(85);CHR$(1);

ESC W Escape W (Double Width)
Format: ESC W;n;
Changes the printer to the Double-Width print mode when ESC W is followed by a 1. This mode is not canceled by a line-feed operation and must be canceled with ESC W followed by a 0 (zero).
Example:
LPRINT CHR$(27);CHR$(87);CHR$(1);

ESC Y Escape Y (960 Bit-Image Graphics Mode Normal Speed)
Format: ESC Y n1;n2;v1;v2;...vk;
Changes from the Text mode to the 960 Bit-Image Graphics mode. The printer prints at normal speed during this operation and cannot print dots on consecutive dot position. The input of data is similar to ESC L.

ESC Z Escape Z (1920 Bit-Image Graphics Mode)

Format: ESC Z;n1;n2;v1;v2;...vk;
Changes from the Text mode to the 1920 Bit-Image Graphics mode. The input is similar to the other Bit-Image Graphics modes. ESC Z can print only every third dot position.

0	1	2	3	4	5	6	7	8	9
NUL							BEL		HT
10	11	12	13	14	15	16	17	18	19
LF		FF	CR	SO	SI			DC2	
20	21	22	23	24	25	26	27	28	29
DC4				CAN			ESC		
30	31	32	33	34	35	36	37	38	39
		SP	!	"	#	$	%	&	'
40	41	42	43	44	45	46	47	48	49
(	)	*	+	,	−	.	/	0	1
50	51	52	53	54	55	56	57	58	59
2	3	4	5	6	7	8	9	:	;
60	61	62	63	64	65	66	67	68	69
<	=	>	?	@	A	B	C	D	E
70	71	72	73	74	75	76	77	78	79
F	G	H	I	J	K	L	M	N	O
80	81	82	83	84	85	86	87	88	89
P	Q	R	S	T	U	V	W	X	Y
90	91	92	93	94	95	96	97	98	99
Z	[	\	]	^	_	`	a	b	c
100	101	102	103	104	105	106	107	108	109
d	e	f	g	h	i	j	k	l	m
110	111	112	113	114	115	116	117	118	119
n	o	p	q	r	s	t	u	v	w
120	121	122	123	124	125	126	127	128	129
x	y	z	{	¦	}	~		NUL	

Printer Character Set 1 (Part 1 of 2)

130	131	132	133	134	135	136	137	138	139
					BEL		HT	LF	
140	141	142	143	144	145	146	147	148	149
FF	CR	SO	SI			DC2		DC4	
150	151	152	153	154	155	156	157	158	159
		CAN			ESC				
160	161	162	163	164	165	166	167	168	169
á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐
170	171	172	173	174	175	176	177	178	179
¬	½	¼	¡	«	»	░	▒	▓	│
180	181	182	183	184	185	186	187	188	189
┤	╡	╢	╖	╕	╣	║	╗	╝	╜
190	191	192	193	194	195	196	197	198	199
╛	┐	└	┴	┬	├	─	┼	╞	╟
200	201	202	203	204	205	206	207	208	209
╚	╔	╩	╦	╠	═	╬	╧	╨	╤
210	211	212	213	214	215	216	217	218	219
╥	╙	╘	╒	╓	╫	╪	┘	┌	█
220	221	222	223	224	225	226	227	228	229
▄	▌	▐	▀	α	β	Γ	π	Σ	σ
230	231	232	233	234	235	236	237	238	239
µ	τ	Φ	Θ	Ω	δ	∞	∅	ϵ	∩
240	241	242	243	244	245	246	247	248	249
≡	±	≥	≤	⌠	⌡	÷	≈	°	▪
250	251	252	253	254	255				
−	√	ⁿ	²	■	SP				

Printer Character Set 1 (Part 2 of 2)

0	1	2	3	4	5	6	7	8	9
NUL			♥	♦	♣	♠	BEL		HT
10	11	12	13	14	15	16	17	18	19
LF		FF	CR	SO	SI			DC2	
20	21	22	23	24	25	26	27	28	29
DC4	§			CAN			ESC		
30	31	32	33	34	35	36	37	38	39
		SP	!	"	#	$	%	&	'
40	41	42	43	44	45	46	47	48	49
(	)	*	+	,	−	.	/	0	1
50	51	52	53	54	55	56	57	58	59
2	3	4	5	6	7	8	9	:	;
60	61	62	63	64	65	66	67	68	69
<	=	>	?	@	A	B	C	D	E
70	71	72	73	74	75	76	77	78	79
F	G	H	I	J	K	L	M	N	O
80	81	82	83	84	85	86	87	88	89
P	Q	R	S	T	U	V	W	X	Y
90	91	92	93	94	95	96	97	98	99
Z	[	\	]	^	_	`	a	b	c
100	101	102	103	104	105	106	107	108	109
d	e	f	g	h	i	j	k	l	m
110	111	112	113	114	115	116	117	118	119
n	o	p	q	r	s	t	u	v	w
120	121	122	123	124	125	126	127	128	129
x	y	z	{	¦	}	~		Ç	ü

Printer Character Set 2 (Part 1 of 2)

130	131	132	133	134	135	136	137	138	139
é	â	ä	à	å	ç	ê	ë	è	ï
140	141	142	143	144	145	146	147	148	149
î	ì	Ä	Â	É	æ	Æ	ô	ö	ò
150	151	152	153	154	155	156	157	158	159
û	ù	ÿ	ö	ü	¢	£	¥	₧	ƒ
160	161	162	163	164	165	166	167	168	169
á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐
170	171	172	173	174	175	176	177	178	179
¬	½	¼	¡	«	»	░	▒	▓	│
180	181	182	183	184	185	186	187	188	189
┤	┤	┤	┐	┐	┤	│	┐	┘	┘
190	191	192	193	194	195	196	197	198	199
┘	┐	└	┴	┬	├	─	┼	├	├
200	201	202	203	204	205	206	207	208	209
└	┌	┴	┬	├	─	┼	┴	┴	┬
210	211	212	213	214	215	216	217	218	219
┬	└	└	┌	┌	┼	┼	┘	┌	█
220	221	222	223	224	225	226	227	228	229
▄	▌	▐	▀	α	β	Γ	π	Σ	σ
230	231	232	233	234	235	236	237	238	239
μ	τ	Φ	Θ	Ω	δ	∞	Ø	ϵ	∩
240	241	242	243	244	245	246	247	248	249
≡	±	≥	≤	⌠	⌡	÷	≈	°	▪
250	251	252	253	254	255				
-	√	ⁿ	²	■	SP				

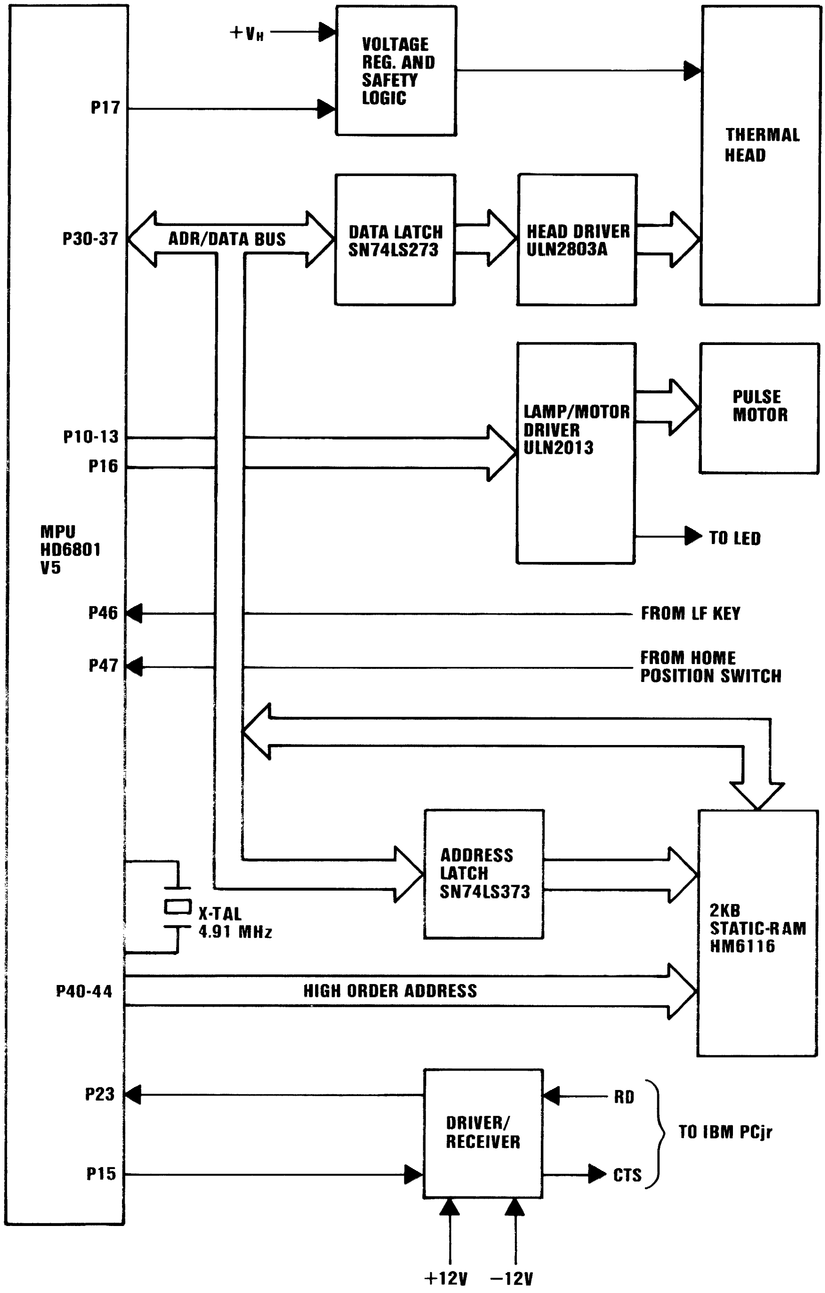
Printer Character Set 2 (Part 2 of 2)

Notes:

IBM PC Compact Printer

The PC Compact Printer is a stand-alone, tabletop unit that plugs into a standard wall outlet. Using an eight-wire print head, the printer can print characters from the standard ASCII, 96-character, uppercase and lowercase character sets, and prints the characters in a 5-by-7 dot matrix at 56 characters-per-second (cps). It prints in one direction (left-to-right) and has four print modes. In the standard mode, the printer prints 80 characters-per-line; in the compressed mode, 136 characters; in the double-width mode, 40 characters, and in the compressed double-width mode, 68 characters-per-line. The PC Compact Printer can also underline characters, has an extended character-set for international languages, and can accept special characters programmed by the user.

The printer has a 1.89 meter (6-foot), 16-lead, printer cable that connects, through an Amphenol connector, to the serial port (RS-232-C) at the rear of the system unit.


+VH
VOLTAGE REG. AND SAFETY LOGIC
P17
THERMAL HEAD
P30-37
ADR/DATA BUS
DATA LATCH SN74LS273
HEAD DRIVER ULN2803A
LAMP/MOTOR DRIVER ULN2013
PULSE MOTOR
P10-13
P16
TO LED
MPU HD6801 V5
P46
FROM LF KEY
P47
FROM HOME POSITION SWITCH
ADDRESS LATCH SN74LS373
2KB STATIC-RAM HM6116
X-TAL 4.91 MHz
P40-44
HIGH ORDER ADDRESS
P23
DRIVER/ RECEIVER
RD
TO IBM PCjr
P15
CTS
+12V
−12V

Printer Specifications

Print Method:	Thermal, non-impact, Dot-matrix
Print Speed:	56 cps
Print Direction:	Left to right only
Number of Pins in Print Head:	8
Line Spacing:	4.23 mm (1/6 in)
Matrix Pattern:	5 by 7 Dots
Character Set:	Full 96-character ASCII with descenders, plus international characters/symbols
Graphics:	None

Print Modes:	Characters per Inch	Maximum Characters per Line
Standard	10	80
Double Width	5	40
Compressed	17.5	136
Compressed/ Double Width	8.75	68

Paper Feed:	Friction Feed
Paper Width:	216 mm (8.5 in)
Copies:	Single sheet only
Paper Path:	Top
System Interface:	Serial Data and Control Lines
Print Color:	Black only

Environmental Conditions	
Temperature:	5°C (+41°F) to 40°C (104°F)
Humidity:	10 to 80% non-condensing
Power Requirement Voltage:	110 Vac 60 Hz
Current:	245 mA
Power Consumption:	36 watts
Heat Output:	57.6 kJ (54.6 BTU)/hr (maximum)
Physical Characteristics	
Height:	88.9 mm (3.5 in)
Width:	312.4 mm (12.3 in)
Depth:	221 mm (8.7 in)
Weight:	2.99 kg (6.6 lb)
Power Cable Length:	1.98 m (6.5 ft)
Size:	28 AWG
Printer Cable Length:	1.83 m (6 ft)
Size:	3 by 18 AWG

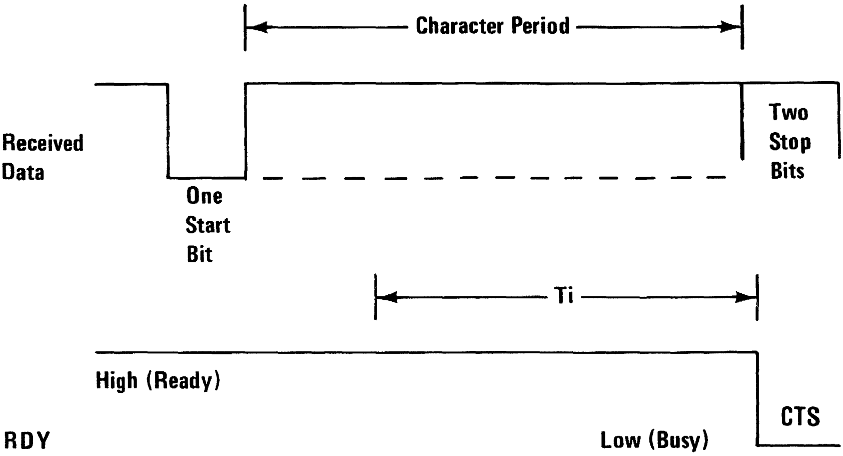
Character Set: ASCII numbers 0 to 31 contain control codes and special characters. ASCII numbers 32 to 127 contain the standard printable characters. ASCII numbers 128 to 175 contain European characters. ASCII numbers 224 to 255 contain math and extra symbols.

Serial Interface Description

Specifications:

Data Transfer Rate:	1200 bps (maximum)
Synchronization:	internal clocking
Handshaking:	CTS (Clear to Send) Pacing
Logic Level:	Input data and all interface control- signals are EIA Levels
Connector Plug:	9804 (Amphenol)

The following figure shows the timing of the Serial Interface.



Serial Interface Timing Diagram

Print Mode Combinations for the PC Compact Printer

The following figure shows the print-mode combinations possible with the PC Compact Printer. Modes shown in the same column can be combined. A print mode can be changed at any time within a line: however, the double-width mode effects the entire line.

Modes					
Standard	XXX				
Compressed		XXX		XXX	XXX
Double-Width			XXX	XXX	XXX
Underline	XXX	XXX	XXX		XXX

Printer Control Codes and Functions

On the following pages you will find a detailed list of the printer control codes and functions. This list also includes descriptions of the functions and examples of the printer control codes.

The examples (LPRINT statements) given in the detailed descriptions of the printer control codes and functions list, are written in BASIC. Some knowledge of BASIC programming is needed to understand these codes. Some of the printer control codes also show a “Format” description when more information is needed for programming considerations.

CODE **PRINTER FUNCTION**

CAN **Cancel**
Clears the printer buffer. Control codes, except SO, remain in effect. Reinitializes the printer to the power on defaults.
LPRINT CHR$(24);

CR **Carriage Return**
Ends the line the printer is on and prints any data remaining in the printer buffer. The logical character position is moved to the left margin. (No Line Feed operation takes place.) Note: IBM Personal Computer BASIC adds a Line Feed unless 128 is added.
LPRINT CHR$(13);

DC2 **Device Control 2 (Compressed Off)**
Stops printing in the Compressed mode.
LPRINT CHR$(18);

DC4 **Device Control 4 (Double Width Off)**
Stops printing in the Double Width mode.
LPRINT CHR$(20);

ESC **Escape**
Informs the printer that the following data is a printer command. (See the following ESC commands.)
LPRINT CHR$(27);

ESC B **Escape B (Set Vertical Tabs)**

Sets vertical tab stop positions. Up to 64 vertical tab stop positions are recognized by the printer. Tab stop positions must be received in ascending numeric order. The tab stop numbers do not become valid until you type the NUL code. Once vertical tab stops are established, they are valid until new tab stops are specified. (If the printer is reset or switched Off, set tab stops are cleared.) If no tab stop is set, the Vertical Tab command acts as a Line Feed command. ESC B followed only by NUL cancels tab stops. The form length must be set by the ESC C command prior to setting tabs.

LPRINT
CHR$(27);CHR$(66);CHR$(10);CHR$(20);
CHR$(40);CHR$(0);

ESC C **Escape C (Set lines per page)**

Format: ESC C;n; Sets the page length. The ESC C command must be followed by a value to specify the length of page desired. (Maximum form length for the printer is 127 lines.) The following example sets the page length to 55 lines. The printer default is 66 lines per page when switched On or reset.

LPRINT CHR$(27);CHR$(67);CHR$(55);

ESC D **Escape D (Set Horizontal Tab Stops)**

Sets the horizontal tab stop positions. The following example shows the horizontal tab stop positions set at printer column positions of 10, 20 and 40. The horizontal tab stops are followed by CHR$(0), the NUL code. They must also be in ascending numeric order as shown. You can set tab stops between 1 and 80. When in the Compressed print mode, you can set tabs up to column 136. The maximum number of tabs that can be set is 112. HT (CHR$(9)) is used to execute a tab operation.

```
LPRINT
CHR$(27);CHR$(68);CHR$(10)CHR$(20)
CHR$(40);CHR$(0);
```

ESC K **Escape K (480 Bit-Image Graphics Mode)**

Format: ESC K;n1;n2; v1; v2;.....vk;
Changes the printer to the Bit-Image Graphics mode. Dot density is 82.5 by 82.5 dots per inch. If the graphics data exceeds the space remaining on the line, the printer ignores the excess data. Only the excess data is lost.

The numbers n1 and n2 specify, in binary form, the number of bit image data bytes to be transferred. Assign values to n1 to represent values from zero to 255 and assign values to n2 to represent values from 0-1 x 256. The total number of bit image data bytes cannot exceed 480. (n1 + (n2 X 256)).

The bit-image data bytes are v1 through vk.

All eight of the print head wires are used to print Bit-image graphics. Each bit of a bit-image data byte represents a dot position within a vertical line. The least significant bit (LSB) represents the bottom dot position, and the most significant bit (MSB) represents the top dot position. For example, if vX is hex 80, the top dot will print only in that vertical position; if vX is hex 01, the bottom dot will print; and if vX is hex FF, all eight dots will print.

	Dot	Bit Number
Top	O - - -	8
	O - - -	7
	O - - -	6
	O - - -	5
	O - - -	4
	O - - -	3
	O - - -	2
Bottom	O - - -	1

LPRINT CHR$(27);CHR$(75);n1;n2

ESC N **Escape N (Set Skip Perforation)**
Format: ESC N;n; Sets the Skip Perforation function. The number following ESC N sets the number of lines to be skipped. The example shows a 12-line skip perforation. This command will print 54 lines and feed the paper 12 lines. The value of n must be between 1 and 127. ESC N must be reset anytime the page length (ESC C) is changed. The default for skip perforation is 25.4 mm (1 inch).
LPRINT CHR$(27);CHR$(78);CHR$(12);

ESC O **Escape O (Cancel Skip Perforation)**
Cancels the Skip Perforation function.
LPRINT CHR$(27);CHR$(79);

ESC R **Escape R (Clear Tabs)**
Resets all tab stops,both horizontal and vertical to the powered-on defaults.
LPRINT CHR$(27);CHR$(82);

ESC W **Escape W (Double Width)**
Format: ESC W;n; Changes the printer to the Double Width mode when ESC W is followed by 1. This mode is not canceled by a line feed operation. It is canceled when ESC W is followed by 0 (zero).
LPRINT CHR$(27);CHR$(87);CHR$(1);

ESC 0 **Escape Zero (1/9-Inch Line Feed)**
Changes the line feed to 2.82 mm (1/9 inch).
LPRINT CHR$(27);CHR$(48);

ESC 1 **Escape One (1/9-inch Line Feed)**
Changes the line feed to 2.82 mm (1/9 inch). ESC 1 functions the same as ESC 0.
LPRINT CHR$(27);CHR$(49);

ESC 2 **Escape Two (Start Variable Line Feeding)**
Resets line spacing to 4.23 mm (1/6 inch). This is the powered-on default for vertical line spacing.
LPRINT CHR$(27);CHR$(50);

ESC 5 **Escape Five (Sets Automatic Line Feed)**
With automatic line feed on, when a CR code is received, a line feed automatically follows after the carriage return. ESC 5 (1) sets auto line feed; ESC 5 (0) resets it.
LPRINT CHR$(27);CHR$(53);

ESC - **Escape Minus (Underline)**
Format: ESC -;n; ESC - followed by 1, prints all of the following data with an underline. ESC - followed by 0 (zero), cancels the Underline print mode.
LPRINT CHR$(27);CHR(45);CHR$(1); [or CHR$(0);]

ESC < **Escape Less Than (Home Head)**
The print head returns to the left margin to print the line following ESC <. This occurs for one line only.
LPRINT CHR$(27);CHR$(60);

FF **Form Feed**
Advances the paper to the top of the next page. Note: The location of the paper, when the printer power switch is set to the On position, determines the top of the page. The next top-of-page is 279 mm (11 inches) from that position. ESC C can be used to change the page length. Always separate multiple Form Feed commands with spaces.
LPRINT CHR$(12);

HT **Horizontal Tab**
Tabs to the next horizontal tab stop. Tab stops are set with ESC D. (Tab stops are automatically set at every 8 columns when the printer power switch is set to the On position.)
LPRINT CHR$(9);

LF **Line Feed**
Advances the paper one line. Line spacing is 4.23 mm (1/6 inch) unless reset by ESC 0, ESC 1, ESC 2.
LPRINT CHR$(10);

NUL **Null**
Used with ESC B and ESC D as terminator for the tab set and clear commands.
LPRINT CHR$(0);

SI **Shift In (Compressed On)**
Changes the printer to the Compressed Character mode. This command is canceled by a DC2 code (Compressed Off).
LPRINT CHR$(15);

SO **Shift Out (Double Width)**
Changes the printer to the Double Width mode. Note: A Carriage Return, Line Feed or DC4 code cancels Double Width mode.
LPRINT CHR$(14);

VT **Vertical Tab**
Spaces the paper to the next vertical tab position. VT are set by the ESC B sequence. The VT command is the same as the LF command, if no tabs are set. The paper is advanced one line after printing or advanced to the next vertical tab stop.
LPRINT CHR$(11);

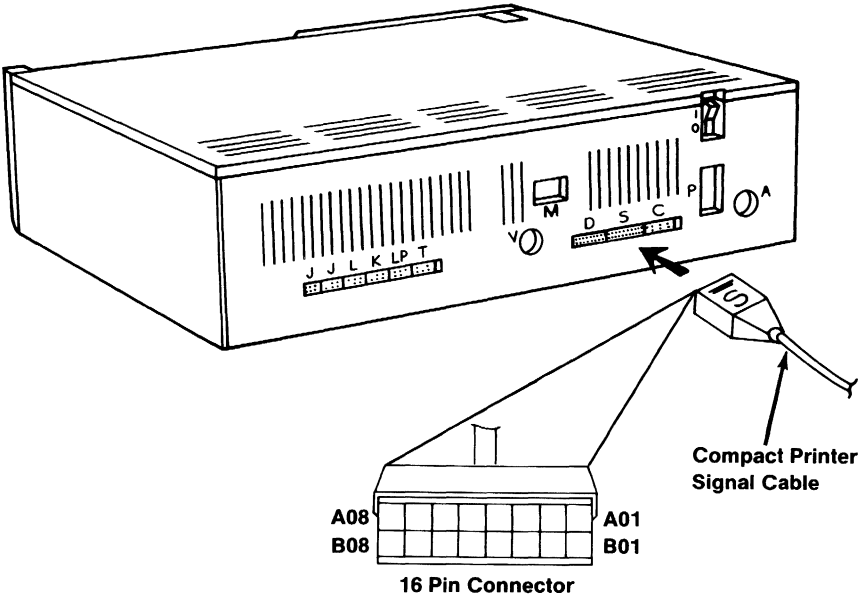
The following charts list the printer control codes and characters in ASCII decimal numeric order, (for example, NUL is 0 and ESC W is 87).

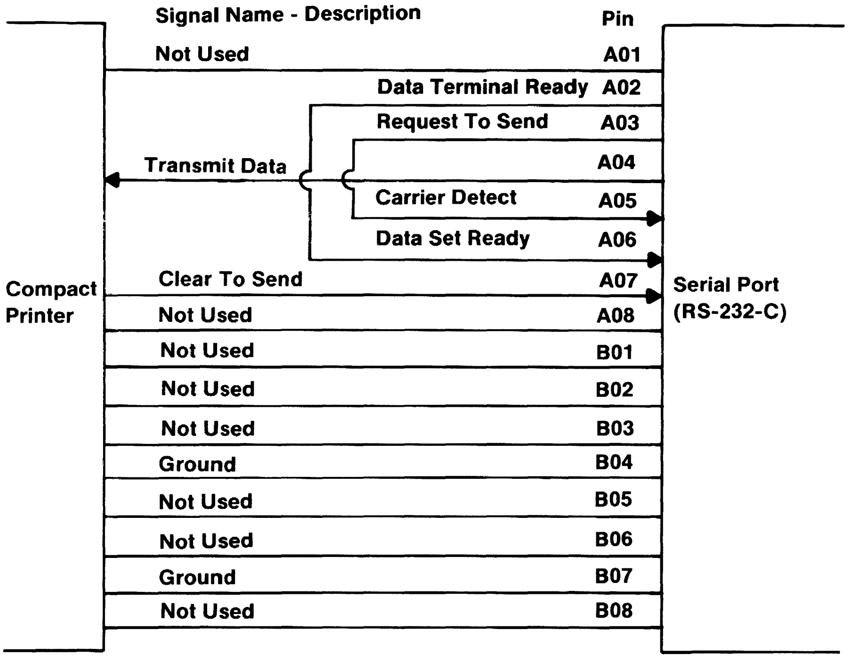
0	1	2	3	4	5	6	7	8	9
NUL			♥	♦	♣	♠	●	◘	HT
10	11	12	13	14	15	16	17	18	19
LF	VT	FF	CR	SO	SI	►	◄	DC2	‼
20	21	22	23	24	25	26	27	28	29
DC4	§	▬	↨	CAN	↓	→	ESC	∟	↔
30	31	32	33	34	35	36	37	38	39
▲	▼	SP	!	"	#	$	%	&	'
40	41	42	43	44	45	46	47	48	49
(	)	*	+	,	—	.	/	0	1
50	51	52	53	54	55	56	57	58	59
2	3	4	5	6	7	8	9	:	;
60	61	62	63	64	65	66	67	68	69
<	=	>	?	@	A	B	C	D	E
70	71	72	73	74	75	76	77	78	79
F	G	H	I	J	K	L	M	N	O
80	81	82	83	84	85	86	87	88	89
P	Q	R	S	T	U	V	W	X	Y
90	91	92	93	94	95	96	97	98	99
Z	[	\	]	^	_	`	a	b	c
100	101	102	103	104	105	106	107	108	109
d	e	f	g	h	i	j	k	l	m
110	111	112	113	114	115	116	117	118	119
n	o	p	q	r	s	t	u	v	w
120	121	122	123	124	125	126	127	128	129
x	y	z	{	¦	}	~	DEL	Ç	ü

Character Set (Part 1 of 2)

130	131	132	133	134	135	136	137	138	139
é	â	ä	à	å	ç	ê	ë	è	ï
140	141	142	143	144	145	146	147	148	149
î	ì	Ä	Â	É	æ	Æ	ô	ö	ò
150	151	152	153	154	155	156	157	158	159
û	ù	ÿ	ö	ü	¢	£	¥	₧	ƒ
160	161	162	163	164	165	166	167	168	169
á	í	ó	ú	ñ	Ñ	a̲	o̲	¿	⌐
170	171	172	173	174	175	176	177	178	179
¬	½	¼	¡	«	»				
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229
				α	β	Γ	π	Σ	σ
230	231	232	233	234	235	236	237	237	239
µ	τ	Φ	Θ	Ω	δ	∞	∅	ϵ	∩
240	241	242	243	244	245	246	247	248	249
≡	±	≥	≤	⌠	⌡	÷	≈	°	■
250	251	252	253	254	255				
-	√	ⁿ	²	■	SP				

Character Set (Part 2 of 2)





Signal Name - Description	Pin
Not Used	A01
Data Terminal Ready	A02
Request To Send	A03
Transmit Data	A04
Carrier Detect	A05
Data Set Ready	A06
Clear To Send	A07
Not Used	A08
Not Used	B01
Not Used	B02
Not Used	B03
Ground	B04
Not Used	B05
Not Used	B06
Ground	B07
Not Used	B08

Data Terminal Ready Looped in Cable to Data Set Ready
Request to Send Looped in Cable to Carrier Detect

Connector Specifications

SECTION 4. COMPATIBILITY WITH THE IBM PERSONAL COMPUTER FAMILY

Contents

Notes:

Compatibility Overview

The IBM PC*jr* is a different Computer than the IBM Personal Computer and IBM Personal Computer XT. Even though it is different, the IBM PC*jr* has a high level of programming compatibility with the IBM Personal Computers. It is possible to create PC*jr* software applications that can run without modification on other IBM Personal Computers. In order to create such programs or to assess if a current program is compatible, you must understand the differences between the Personal Computers in the IBM family and know the proper way to communicate with them.

Normally, it would be impossible for a program written for one computer to run on a different computer since the microprocessors would be different; and the language of the application could not be executed by different processors. In this case, the application would have to be re-written entirely in the language of the other processor. Since the IBM PC*jr* and the other IBM Personal Computers use exactly the same microprocessors (Intel 8088), most assembler language programs need not be modified.

This alone is not enough, since applications normally take advantage of a computers device services (BIOS) and operating system (IBM DOS 2.1). In order to allow for maximum program compatibility, the IBM PC*jr* has maintained all BIOS system interrupts and utilizes the same IBM DOS. This means that applications which use the BIOS and the IBM DOS interrupts on the IBM Personal Computers operate the same on the IBM PC*jr*.

Note: The BIOS micro-code of the IBM PC*jr* is not identical to that of the IBM Personal Computers. If an application bypasses the BIOS interrupt calls and

directly accesses routines and/or storage locations in one system, it may not run in the other system. Some routines may be similar and some BIOS storage locations may be the same. It is strongly recommended that applications use only the BIOS and DOS interrupt interfaces in order to achieve compatibility in the IBM Personal Computer family.

Using the same language and the BIOS and DOS interfaces go a long way in achieving application compatibility. However, there are still several factors which need to be taken into consideration:

- Timing Dependencies

- Unequal Configurations

- Hardware Differences

Timing Dependencies

Programs running in user read/write memory normally run slower on the PC*jr* than on the IBM Personal Computers. Programs running in read-only memory (ROM) normally run a little faster on the PC*jr* than on the IBM Personal Computers. This may or may not cause a difference depending upon the application. Most applications are very I/O dependent in which case the execution time is not the critical factor and may not be noticeable. In other cases, the application runs the same but merely take a different amount of time.

If an application has very critical timing dependencies, any timing differences (faster or slower) may adversely affect its usability. Using an application's program execution speed to achieve a desired timing can effect the application. In these cases, the application may need to be modified.

> **Note:** It is strongly recommended not to depend on instruction execution speed to achieve specific application timing. The system timer can provide short interval timing for assembly language programs. Similar timing functions are available in BASIC.

Performance of specific I/O devices (such as diskette or printer) may also differ between the PC*jr* and the other IBM Personal Computers. You should also avoid using timing of any I/O device as a dependency for the application.

Notes:

Unequal Configurations

In designing an application to run on both the IBM PC*jr* and the IBM Personal Computers, you need to make sure that the required hardware configuration is available on all machines. This means the application's minimum requirements are met by all IBM Personal Computers.

Notes:

Hardware Differences

To be able to run on either computer without change, an application utilizing a specific I/O device must have access to identical devices (or devices with identical operating characteristics and interfaces). The IBM PC*jr* and the IBM Personal Computers have very compatible I/O device capabilities.

The following table lists the hardware features and I/O devices supported by the IBM PC*jr* and the IBM Personal Computers and summarizes the differences:

Device	PC	PCXT	PC*jr*	PC*jr* Comments
Maximum User Memory	640KB	640KB	128KB	Shares user RAM with Video Buffer
Cordless Keyboard	No	No	Yes	Scan codes compatible and full 83 key capability
83 Key Keyboard	Yes	Yes	No	Compatible, but Hardware interface differences
Diskette Drive	Yes	Yes	Yes	Compatible, but different address and no DMA support
Hard Disk File	No	Yes	No	
Parallel Printer	Yes	Yes	Yes	Compatible
RS 232 Serial Port	Yes	Yes	Yes	Compatible, hex 2F8 address, Interrupt Level 3, Baud-Rate-Frequency divisor difference
Game Control	Yes	Yes	Yes	Compatible interface with potential timing differences
Cassette	Yes	No	Yes	Compatible
Internal Modem	No	No	Yes	Compatible to PC Serial Port hex 3F8 address, Interrupt Level 4, frequency divisor difference
IBM Monochrome Display	Yes	Yes	No	
Color Graphics and Display	Yes	Yes	Yes	Compatible, with some register differences and enchancements
Light Pen	Yes	Yes	Yes	Compatible

PC*jr* and Personal Computers Comparison (Part 1 of 2)

Device	PC	PCXT	PC*jr*	PC*jr* Comments
Attachable Joystick	Yes	Yes	Yes	Compatible
8253 Timer (time of day)	Yes	Yes	Yes	Compatible
8259 Interrupt	Yes	Yes	Yes	Some difference in interrupt levels
Internal Sound	Yes	Yes	Yes	Compatible but less frequency response
TI 76496 Sound	No	No	Yes	
ROM Cartridge Interface	No	No	Yes	
Future I/O ROM Architecture	Yes	Yes	Yes	Compatible

PC*jr* and Personal Computers Comparison (Part 2 of 2)

The hardware differences between the IBM PC*jr* and the IBM Personal Computers may lead to incompatibilities depending upon the specific application. Once again; if your application maintains an interface to the Personal Computer Family at the BIOS and DOS interrupt levels, then all hardware differences are handled transparently to your application. If your application goes below the BIOS level and directly addresses the hardware, then there could be an incompatibility.

User Read/Write Memory

Memory difference can be a problem even with programs written for the same computer, if the available memory is not the same from one machine to the next. Thus, the deciding factor is to state what the minimum memory requirement is for the application, and require that amount on the computer in question.

It is important to understand the memory aspects of the IBM PC*jr* in relationship to that of the IBM Personal Computers. The IBM PC*jr* can be configured for 64K bytes or 128K bytes (with memory expansion). However, this user memory is not all available to the application. The IBM PC*jr* video architecture utilizes a minimum of 16K bytes (in graphic mode) and 2K bytes (in alpha numeric mode) for the screen buffer. Therefore (in graphics mode), the IBM PC*jr* really has 48K bytes or 112K bytes (with memory expansion) available for system software. This is not the case with the IBM Personal Computers, since the color graphics adapter contains a separate 16K byte screen buffer. Thus, a 64K bytes Personal Computer with color graphics (extra 16K bytes) is an 80K byte system compared to a 64K byte IBM PC*jr*. The IBM PC*jr* also has graphic enhancements which allow more than the 16K bytes to be utilized for video screen buffers. If these enhanced features are used in an application, then even less is available for user memory.

Another aspect of available memory is the amount taken away by operating systems and language interpreters. In the case of the IBM DOS, both the IBM PC*jr* and the IBM Personal Computers support the same DOS. If your application requires the BASIC interpreter, then there may be a difference. The IBM Personal Computer Cassette BASIC resides entirely in the system ROM; taking no user memory. However, Disk BASIC or Advanced BASIC utilizes approximately 10K bytes and 14K bytes respectively from user memory. In the IBM PC*jr*, Advanced BASIC capabilities (cartridge BASIC) reside in ROM, taking no user memory.

As you can see, many items factor into user available memory requirements. The most frequent comparison is for the assembler language or compiled application using a 16K-byte screen buffer operating under DOS 2.1. In this case, an application requiring 64K bytes of user memory on an IBM Personal Computer cannot run on the IBM PC*jr* without its expansion memory (128K byte capability). This is because of the IBM PC*jr* video usage of 16K bytes. Also, any application requiring more than 112K bytes of user memory with DOS 2.1 on the IBM Personal Computers cannot run on an IBM PC*jr*.

Diskette Capacity/Operation

Since the IBM PC*jr* maximum stand-alone configuration is one diskette drive with a maximum capacity of 360K bytes diskette storage , an IBM PC*jr* application is either limited by this diskette capacity or is impacted by the user having to change diskettes more frequently. The IBM Personal Computers can have multiple diskette drives with a capacity of 360K bytes diskette storage each or even possess hard files with a much larger disk storage capacity. This capacity difference may or may not be a concern depending upon the specific application.

In terms of diskette interfacing, the IBM PC*jr* and the IBM Personal Computers both utilize the NEC μPD765 floppy diskette controller, but with different hardware addresses, and the IBM PC*jr* does not operate through direct memory access (DMA). Since the IBM PC*jr* does not have DMA capability, application programs cannot overlap diskette I/O operations. When diskette I/O takes place, the entire system is masked (operator keystrokes and asynchronous communications cannot take place). Therefore, the application must insure that asynchronous operations do not take place while diskette I/O is active.

IBM PC*jr* Cordless Keyboard

The Cordless Keyboard is unique to the IBM PC*jr*. Even though it does not possess all 83 keys of the IBM Personal Computers' keyboards, it does have the capability to generate all of the scan codes of the 83-key keyboard.

The following shows the additional functions available on the PC*jr*.

PC*jr* Special Functions	Required Key Combinations
Shift screen to the left	Alt + Ctrl + cursor left
Shift screen to the right	Alt + Ctrl + cursor right
Audio Feedback (System clicks when a key is pressed.	Alt + Ctrl + Caps Lock
Customer Diagnostics	Alt + Ctrl + Ins

PC*jr* Special Functions

For more detail see "Keyboard Encoding and Usage" in Section 5.

Since all scan codes can be generated, any special application requirements can be met on the Cordless Keyboard.

The highest level of compatibility to interface to keyboards is through BIOS Interrupt hex 16 (read keystroke). Below that level is risky since there are hardware differences between the PC*jr* keyboard and the IBM Personal Computers' keyboards. The PC*jr* system utilizes the non-maskable (NMI) Interrupt to deserialize the scan codes and pass it to Interrupt hex 48 for compatible mapping to 83-key format. Interrupt level 9 remains a compatible interface for 83-key scan-code handling. It is not recommended to replace Interrupt level 9 even though a high degree of compatibility is maintained. If necessary, analyze this architecture carefully.

Color Graphics Capability

The IBM PC*jr* color graphic architecture is quite different from that of the IBM Personal Computers. The main difference (as previously discussed) is that the video buffer is taken from main user memory rather than having separate memory for video (as in the IBM Personal Computers). Normally, this would be an incompatibility since applications directly address the color graphics buffer at hex B8000. However, the IBM PC*jr* has special hardware to redirect hex B8000 addressing to any specific 16K-byte block of its user memory. The IBM PC*jr* defaults the video buffer to the high end 16K-byte block of user memory and applications can continue to address the video buffer at hex B8000. In addition all IBM Personal Computers' color graphics adapter modes are BIOS compatible and memory structure (bit map) compatible. These modes are:

Modes	Requirements
Alphanumeric:	
40x25 BW	None
40x25 Color	None
80x25 Color	Note
80x25 BW	None
Graphics:	
320x200 4 Color	None
320x200 BW	None
640x200 BW	None
Note: PC*jr* requires the 64KB Memory and Display Expansion.	

Modes Available on the IBM Personal Computers and PC*jr*

In addition the IBM PC*jr* provides some new enhanced graphic modes which are not available to the IBM Personal Computers.

Modes	Requirements
Graphics:	
320x200 16 Color	Note
640x200 4 Color	Note
160x200 16 Color	None
Note: PC*jr* requires the 64KB Memory and Display Expansion.	

Modes Available Only on PC*jr*

The IBM PC*jr* and IBM Personal Computers utilize the 6845 controller, but the hardware interface is not completely the same. Hardware addresses hex 3D8 and hex 3D9 are not supported by the IBM PC*jr* video interface. Requests using these two addresses are not honored.

Also there are differences in the actual video used by the hardware. BIOS maintains compatibility by using the appropriate PC*jr* video parameters (addressed through Interrupt hex 1D) and maintains all video calls (through Interrupt hex 10). Application can still specify video parameter overrides by modifying Interrupt hex 1D to address their own parameters; however, since there are hardware differences the recommended approach is as follows:

1. Copy the original parameters from the BIOS of the system.
2. Change only those parameters desired.
3. Consider the specific video differences between systems.

Other differences to be aware of are:

- The IBM PC*jr* defaults the colorburst mode to be off, whereas the IBM Personal Computers default colorburst to on. Thus applications should not assume either default but set colorburst mode (through BIOS call) to the desired setting.
- The IBM PC*jr* video supports a full gray scale capability which the IBM Personal Computers do not.
- There can be some color differences between the IBM Personal Computers and the IBM PC*jr*; especially when color mixing techniques are used.

Black and White Monochrome Display

The IBM PC*jr* does not support the IBM Personal Computers black and white monochrome display. Programs which directly address the IBM Personal Computers monochrome display are not compatible. For example, any direct addressing of the B&W video buffer at hex B8000 is not redirected by the IBM PC*jr*. Applications should support Personal Computer video capabilities through BIOS, and the video buffer address is either transparent to the application or the address is provided indirectly in the BIOS data area.

RS232 Serial Port and IBM PC*jr* Internal Modem

The IBM PC*jr* serial port address is hex 2F8 and is associated with hardware Interrupt level 3. This is compatible with a second Asynchronous Communications Adapter on the IBM Personal Computers. The Internal Modem address is hex 3F8 and is associated with Interrupt level 4. This is compatible with the first Asynchronous Communications Adapter on the IBM Personal Computers. It is important to note that when the IBM PC*jr* has the Internal Modem installed it is logically COM1 and the RS232 serial port is logically COM2 in BIOS, DOS, and BASIC. Without the Internal Modem installed the RS232 serial port is logically addressed as COM1 in BIOS, DOS, and BASIC even though its address is still hex 2F8 using Interrupt level 3. Other hardware differences on the PC*jr* serial devices are:

- A different frequency divisor is needed to generate baud rate. This is transparent to applications using BIOS to initialize the devices (Interrupt Hex 14).
- No ring indicate capability on the RS232 serial port.

- Asynchronous communications input cannot be overlapped with IBM PC*jr* diskette I/O. Since diskette I/O operates in a non-DMA mode any asynchronous data received during diskette activity may be overrun (and lost). Thus, applications must insure that no diskette activity is active while receiving asynchronous communication data. This can be done by pacing the asynchronous device (tell it to hold from sending). The ASCII characters XOFF and XON are frequently used by some host computers for this purpose.

Summary

In summary, the IBM PC*jr* is a member of the IBM Personal Computer family by way of its strong architecture compatibility. The highest degree of application compatibility can be achieved by using a common high level language, and/ or accessing the system only through BIOS and DOS interrupts. It's not recommended to go below the BIOS level even though there are other hardware compatibilities. When it is necessary to design for particular computer differences, the application should determine at execution time which particular computer it is running on. This can be done by inspecting the ROM memory location at segment address hex F000 and offset hex FFFE for the following values

hex FF	= the IBM Personal Computer
hex FE	= the IBM Personal Computer XT
hex FD	= the IBM PC*jr*

Once determined, dual paths would handle any differences.

Notes:

SECTION 5. SYSTEM BIOS USAGE

Contents

ROM BIOS

The basic input/output system (BIOS) resides in ROM on the system board and provides device-level control for the major I/O devices in the system. Additional ROM modules may be located on option adapters to provide device level control for that option adapter. BIOS routines enable the assembly-language programmer to perform block (diskette) or character-level I/O-operations without concern for device address and operating characteristics. System services, such as time-of-day and memory-size determination, are provided by the BIOS.

The goal is to provide an operational interface to the system and relieve the programmer of the concern about the characteristics of hardware devices. The BIOS interface insulates the user from the hardware, allowing new devices to be added to the system, yet retaining the BIOS-level interface to the device. In this manner, user programs become transparent to hardware modifications and enhancements.

The IBM Personal Computer *Macro Assembler* manual and the IBM Personal Computer *Disk Operating System* (DOS) manual provide useful programming information related to this section.

Notes:

BIOS Usage

Access to BIOS is through the software interrupts. Each BIOS entry-point is available through its own interrupt, which can be found in “Personal Computer BIOS Interrupt Vectors”, later in this section.

The software interrupts, hex 10 through hex 1A, each access a different BIOS-routine. For example, to determine the amount of memory available in the system,

INT hex 12

invokes the BIOS routine for determining memory size and returns the value to the caller.

All parameters passed to and from the BIOS routines go through the 8088 registers. The prologue of each BIOS function indicates the registers used on the call and the return. For the memory size example, no parameters are passed. The memory size, in 1K byte increments, is returned in the AX register.

If a BIOS function has several possible operations, the AH register is used at input to indicate the desired operation. For example, to set the time-of-day, the following code is required:

```
MOV  AH,1                ;function is to set time-of-day.
MOV  CX,HIGH_COUNT        ;establish the current
MOV  DX,LOW_COUNT
INT  1AH                 ;set the time.
```

To read time-of-day:

```
MOV   AH,0               ;function is to read time of day.
INT   1AH                ;read the timer.
```

Generally, the BIOS routines save all registers except for AX and the flags. Other registers are modified on return, only if they are returning a value to the caller. The exact register usage can be seen in the prologue of each BIOS function.

Address (Hex)	Interrupt Number	Name	BIOS Entry
0-3	0	Divide by Zero	D_EOI
4-7	1	Single Step	D_EOI
8-B	2	Keyboard NMI	KBDNMI
C-F	3	Breakpoint	D_EOI
10-13	4	Overflow	D_EOI
14-17	5	Print Screen	PRINT_SCREEN
18-1B	6	Reserved	D_EOI
1D-1F	7	Reserved	D_EOI
20-23	8	Time of Day	TIMER_INT
24-27	9	Keyboard	KB_INT
28-2B	A	Reserved	D_EOI
2C-2F	B	Communications	D_EOI
30-33	C	Communications	D_EOI
34-37	D	Vertical retrace	D_EOI
38-3B	E	Diskette Error Handler	DISK_INT
3C-3F	F	Printer	D_EOI
40-43	10	Video	VIDEO_IO
44-47	11	Equipment Check	EQUIPMENT
48-4B	12	Memory	MEMORY_SIZE_DETERMINE
4C-4F	13	Diskette	DISKETTE_IO
50-53	14	Communications	RS232_IO
54-57	15	Cassette	CASSETTE_IO
58-5B	16	Keyboard	KEYBOARD_IO
5C-5F	17	Printer	PRINTER_IO
60-63	18	Resident BASIC	F600:0000
64-67	19	Bootstrap	BOOT_STRAP
68-6B	1A	Time of Day	TIME_OF_DAY
6C-6F	1B	Keyboard Break	DUMMY_RETURN
70-73	1C	Timer Tick	DUMMY_RETURN
74-77	1D	Video Initialization	VIDEO_PARMS
78-7B	1E	Diskette Parameters	DISK_BASE
7C-7F	1F	Video Graphics Chars	CRT_CHARH

Personal Computer BIOS Interrupt Vectors

Vectors with Special Meanings

The following are vectors with special meanings.

Interrupt Hex 1B - Keyboard Break Address

This vector points to the code to be executed when **Break** is pressed on the keyboard. The vector is invoked while responding to the keyboard interrupt, and control should be returned through an IRET instruction. The POWER-ON routines initialize this vector to an IRET instruction, so that nothing occurs when **Break** is pressed unless the application program sets a different value.

Control may be retained by this routine, with the following problem. The 'Break' may have occurred during interrupt processing, so that one or more 'End of Interrupt' commands must be issued in case an operation was underway at that time.

Interrupt Hex 1C - Timer Tick

This vector points to the code to be executed on every system-clock tick. This vector is invoked while responding to the 'timer' interrupt, and control should be returned through an IRET instruction. The POWER-ON routines initialize this vector to point to an IRET instruction, so that nothing occurs unless the application modifies the pointer. It is the responsibility of the application to save and restore all registers that are modified.

Interrupt Hex 1D - Video Parameters

This vector points to a data region containing the parameters required for the initialization of the 6845 CRT Controller. Note that there are four separate tables, and all four must be reproduced if all modes of operation are to be supported. The POWER-ON routines initialize this vector to point to the parameters contained in the ROM video-routines. It is recommended that if a programmer wishes to use a different parameter table, that the table contained in ROM be copied to RAM and just modify the values needed for the application.

Interrupt Hex 1E - Diskette Parameters

This vector points to a data region containing the parameters required for the diskette drive. The POWER-ON routines initialize the vector to point to the parameters contained in the ROM DISKETTE-routine. These default parameters represent the specified values for any IBM drives attached to the machine. Changing this parameter block may be necessary to reflect the specifications of the other drives attached. It is recommended that if a programmer wishes to use a different parameter table, that the table contained in ROM be copied to RAM and just modify the values needed for the application. The motor start-up-time parameter (parameter 10) is overridden by BIOS to force a 500-ms delay (value 04) if the parameter value is less than 04.

Interrupt Hex 1F and hex 44 - Graphics Character Pointers

When operating in the graphics modes, the read/write-character interface forms the character from the ASCII code-point, using a table of dot patterns where each code point is comprised of 8 bytes of graphics information. The table of dot patterns for the first 128 code-points contained in ROM is pointed to by Interrupt Hex 44 and the second table of 128 code-points contained in ROM is pointed to by Interrupt Hex 1F. The user can change this vector to point to his own table of dot patterns. It is the responsibility of the user to restore these vectors to point to the default code-point-tables at the termination of the program.

Interrupt Hex 48 - Cordless Keyboard Translation

This vector points to the code responsible for translating keyboard scan-codes that are specific to the Cordless Keyboard. The translated scan-codes are then passed to the code pointed to by Interrupt Hex 9 which then handles the 83-key Keyboard scan codes.

Interrupt Hex 49 - Non-Keyboard Scan-Code Translation-Table Address

This interrupt contains the address of a table used to translate non-keyboard scan-codes (scan codes greater than 85 excluding 255.) If Interrupt hex 48 detects a scan code greater than 85 (excluding 255) it translates it using the table pointed to by Interrupt Hex 49. The address that Interrupt Hex 49 points to can be changed by users to point to their own table if different translations are required.

Note: It is recommended that a programmer save default pointers and restore them to their original values when the program has terminated.

Notes:

Other Read Write Memory Usage

The IBM BIOS routines use 256 bytes of memory starting at absolute hex 400 to hex 4FF. Locations hex 400 to 407 contain the base addresses of any RS-232C attachments to the system. This includes the optional IBM PC*jr* Internal Modem and the standard RS232 serial-port. Locations hex 408 to 40F contain the base addresses of any parallel printer attachments.

Memory locations hex 300 to 3FF are used as a stack area during the power-on initialization, and bootstrap, when control is passed to it from power-on. If the user desires the stack in a different area, the area must be set by the application.

The following is a list of the interrupts reserved for BIOS, DOS, and BASIC.

Address (Hex)	Interrupt (Hex)	Function
80-83	20	DOS Program Terminate
84-87	21	DOS Function Call
88-8B	22	DOS Terminate Address
8C-8F	23	DOS Ctrl Break Exit Address
90-93	24	DOS Fatal Error Vector
94-97	25	DOS Absolute Disk Read
98-9B	26	DOS Absolute Disk Write
9C-9F	27	DOS Terminate, Fix in Storage
A0-FF	28-3F	Reserved for DOS
100-115	40-43	Reserved for BIOS
116-119	44	First 128 Graphics Characters
120-131	45-47	Reserves for BIOS
132-135	48	Cordless-Keyboard Translation
136-139	49	Non-keyboard Scan-code Translation Table
140-17F	50-5F	Reserved for BIOS
100-17F	40-5F	Reserved for BIOS
180-19F	60-67	Reserved for User Software Interrupts
1A0-1FF	68-7F	Reserved
200-217	80-85	Reserved for Basic
218-3C3	86-F0	Used by Basic Interpreter while BASIC is running
3C4-3FF	F1-FF	Reserved

BIOS, BASIC, and DOS Reserved Interrupts

The following is a list of reserved memory locations.

Address (Hex)	Mode	Function
400-48F	ROM BIOS	See BIOS Listing
490-4EF		Reserved for System Usage
500-5FF		Communication Area for any application
500	DOS	Reserved for DOS and BASIC, Print Screen Status Flag Store, O-Print Screen Not Active or Successful Print Screen Operation, 1-Print Screen In Progress, 255-Error Encountered During Print Screen Operation,
504	DOS	Single Drive Mode Status Byte
510-511	BASIC	BASIC's segment Address Store
512-515	BASIC	Clock Interrupt Vector Segment: Offset Store
516-519	BASIC	Break key Interrupt Vector Segment: Offset Store
51A-51D	BASIC	Disk Error Interrupt Vector Segment: Offset Store

Reserved Memory Locations

The following is a list of the BASIC workspace variables.

If you do DEF SEG (Default workspace segment):	Offset (Hex)	Length
Line number of current line being executed	2E	2
Line number of last error	347	2
Offset into segment of start of program text	30	2
Offset into segment of start of variables (end of program text 1-1)	358	2
Keyboard buffer contents if 0-no characters in buffer if 1-characters in buffer	6A	1
Character color in graphics mode Set to 1, 2, or 3 to get text in colors 1 to 3. Do not set to 0. (Default = 3)	4E	1

Example

```
100 Print Peek (&H2E) + 256*Peek (&H2F)
 )        L          H
 (
100  | hex 64 | hex 00 |
```

BASIC Workspace Variables

The following shows the mapping of the BIOS memory

Starting Address in Hex	
00000	BIOS Interrupt Vectors
00400	BIOS Data Area
00500	User Read/Write Memory
A0000	Reserved for Future Video
B8000	Reserved for Video
C0000	Reserved for Future I/O ROM
D0000	Reserved for Cartridges
E0000	Reserved for Cartridges
F0000	BIOS/ Diagnostics/ Cassette and BASIC Program Area

BIOS System Map

BIOS Programming Guidelines

The BIOS code is invoked through software interrupts. The programmer should not 'hard code' BIOS addresses into applications. **The internal workings and absolute addresses within BIOS are subject to change without notice.**

If an error is reported by the diskette code, you should 'reset' the drive adapter and retry the operation. A specified number of retries should be required on diskette 'reads' to insure the problem is not due to motor start-up.

When altering I/O-port bit-values, the programmer should change only those bits which are necessary to the current task. Upon completion, the programmer should restore the original environment. Failure to adhere to this practice may be incompatible with present and future systems.

Adapter Cards with System-Accessible ROM-Modules

The ROM BIOS provides a facility to integrate adapter cards with on-board ROM-code into the system. During the Power-On Self-Test (POST), interrupt vectors are established for the BIOS calls. After the default vectors are in place, a scan for additional ROM modules takes place. At this point, a ROM routine on the adapter card may gain control. The routine may establish or intercept interrupt vectors to hook themselves into the system.

The absolute addresses hex C0000 through hex D0000 are scanned in 2K-byte blocks in search of a valid adapter card ROM. A valid ROM is defined as follows:

Byte 0:	hex 55
Byte 1:	hex AA
Byte 2:	length (multiple of 2K bytes) - A length indicator representing the number of 512-byte blocks in the ROM (length/512). A checksum is also done to test the integrity of the ROM module. Each byte in the defined ROM is summed modulo hex 100. This sum must be 0 for the module to be deemed valid.

When the POST identifies a valid ROM, it does a 'far call' to byte 3 of the ROM (which should be executable code). The adapter card may now perform its power-on initialization-tasks. The feature ROM should return control to the BIOS routines by executing a 'far return'.

Notes:

Keyboard Encoding and Usage

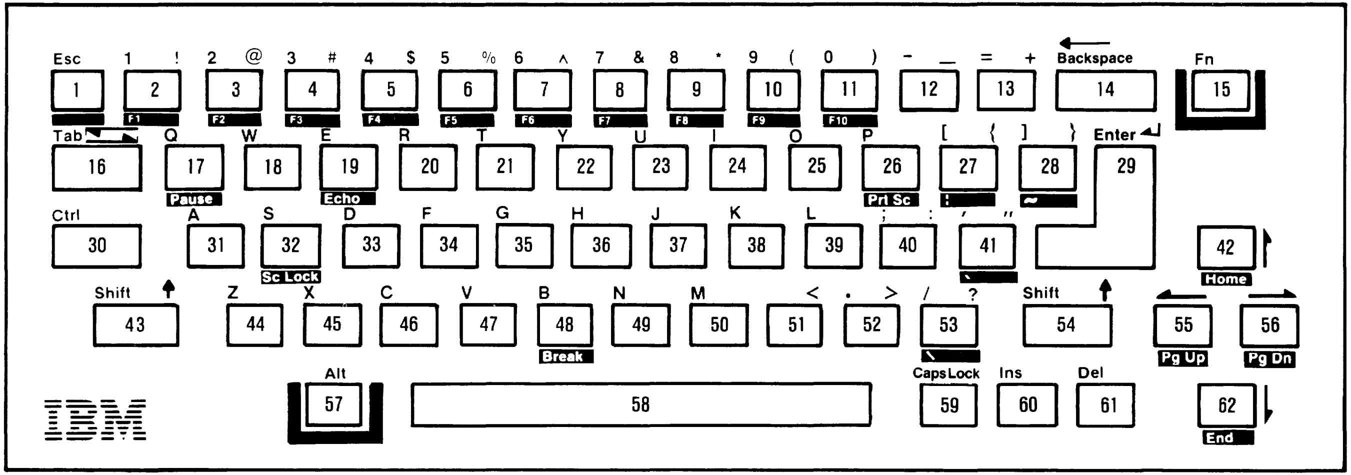
The following explains how the keyboard interacts with BIOS and how 83-key-keyboard functions are accomplished on the Cordless Keyboard.

Cordless Keyboard Encoding

The KEYBOARD routine provided by IBM in the ROM BIOS is responsible for converting the keyboard scan-codes into what is termed "Extended ASCII."

Extended ASCII encompasses one-byte character-codes with possible values of 0 to 255, an extended code for certain extended keyboard-functions, and functions handled within the KEYBOARD routine or through interrupts.

The following is the physical layout of the IBM PC*jr* Cordless Keyboard.



IBM PC*jr* Cordless Keyboard Diagram

The following are charts of the scan codes for the IBM PC*jr* Cordless Keyboard.

Key Position	**Keyboard Characters**	**Make Code (Hex)**	**Break Code (Hex)**
1	ESC	1	81
2	1/!	2	82
3	2/∂	3	83
4	3/#	4	84
5	4/$	5	85
6	5/%	6	86
7	6/≙	7	87
8	7/&	8	88
9	8/*	9	89
10	9/(	A	8A
11	0/)	B	8B
12	-/—	C	8C
13	=/+	D	8D
14	BS<—	E	8E
15	FN	54	D4
16	TAB	F	8F
17	q/Q	10	90
18	w/W	11	91
19	e/E	12	92
20	r/R	13	93
21	t/T	14	94
22	y/Y	15	95
23	u/U	16	96
24	i/I	17	97
25	o/O	18	98
26	p/P	19	99
27	[/{	1A	9A
28	]/}	1B	9B
29	ENTER	1C	9C
30	CTRL	1D	9D
31	a/A	1E	9E

Cordless Keyboard Maxtrix Scan Codes (Part 1 of 2)

Key Position	Keyboard Characters	Make Code (Hex)	Break Code (Hex)
32	s/S	1F	9F
33	d/D	20	A0
34	f/F	21	A1
35	g/G	22	A2
36	h/H	23	A3
37	j/J	24	A4
38	k/K	25	A5
39	l/L	26	A6
40	;/:	27	A7
41	’/”	28	A8
42	CUR.UP	48	C8
43	LF.SHIFT	2A	AA
44	z/Z	2C	AC
45	x/X	2D	AD
46	c/C	2E	AE
47	v/V	2F	AF
48	b/B	30	B0
49	n/N	31	B1
50	m/M	32	B2
51	,/<	33	B3
52	./>	34	B4
53	//?	35	B5
54	RT.SHIFT	36	B6
55	CUR.LF.	4B	CB
56	CUR.RT.	4D	CD
57	ALT.	38	B8
58	SP.BAR	39	B9
59	CAPS LOCK	3A	BA
60	INSERT	52	D2
61	DELETE	53	D3
62	CUR.DWN.	50	D0
Phantom-Key Scan Code		55	

Cordless Keyboard Matrix Scan Codes (Part 2 of 2)

The Cordless Keyboard is unique to the PC*jr*. Even though it does not possess all 83 keys of the IBM Personal Computer keyboard, it does have a way in which you can cause all of the scan codes of the 83-key keyboard. The following chart shows the mapping of functions between both keyboards:

IBM Personal Computers 83-key Keyboard Function	**IBM PC*jr* Cordless Keyboard Mapping**
F1-F10	Function key + 1-0 (F1-F10)
Ctrl Break	Function key + B (Break)
Ctrl PrtSc (Echo Print)	Function key + E (Echo)
Shift PrtSc (Print Screen)	Function key + P (PrtSc)
Ctrl NumLock (Pause)	Function key + Q (Pause)
Scroll Lock	Function key + S (ScLock)
Numeric keypad region:	
Num Lock (Number keypad 1 through 10 becomes key scan codes.)	Alt + Function key + N (1 through 0 becomes numeric-key scan-codes)
PgUp key	Function key + cursor left (PgUp)
PgDn key	Function key + cursor right (PgDn)
Home key	Function key + cursor up (Home)
End key	Function key + cursor down (End)
Numeric keypad - sign	Function key plus the - sign
Numeric keypad + sign	Function key + = sign
\ key	Alt + /
' key	Alt + '
! key	Alt + [
~ key	Alt +]
* with PrtSc	Alt + .
Numeric keypad .	Shift + Del
All 256 extended codes: Alt + numeric value from numeric keypad	NumLock then Alt + numeric value (1 through 0)

83-key-Keyboard Function to Cordless-Keyboard Mapping

Character Codes

The following character codes are passed through the BIOS KEYBOARD-routine to the system or application program. A **-1** means the combination is suppressed in the KEYBOARD routine. The codes are returned in AL. See Appendix C, "Characters, Keystrokes,and Color" for the exact codes.

Key Number	Base Case	Upper Case	Ctrl	Alt	Fn
1	Esc	Esc	Esc	-1	**
2	1	!	-1	*,*****	(F1) *,***
3	2	∂	Nul (000)	*,*****	(F2) *,***
4	3	#	-1	*,*****	(F3)
5	4	$	-1	*,*****	(F4) *,***
6	5	%	-1	*,*****	(F5) *,***
7	6	≙	RSO (030)	*,*****	(F6) *,***
8	7	&	-1	*,*****	(F7) *,***
9	8	*	-1	*,*****	(F8) *,***
10	9	(	-1	*,*****	(F9) *,***
11	0	)	-1	*,*****	(F10) *,***
12	—	-	US (031)	*	***
13	=	+	-1	*	***
14	Backspace (008)	Backspace (008)	DEL (127)	-1	-1
15 Fn	-1	-1	-1	-1	-1
16	—>| (009)	|<— *	-1	-1	-1
17	q	Q	DC1 (017)	*	**,*** (Pause)
18	w	W	ETB (023)	*	-1
19	e	E	ENQ (005)	*	**,*** (Echo)
20	r	R	DC2 (018)	*	-1
21	t	T	DC4 (020)	*	-1

* - Refer to "Extended Codes" in this section.

** - Refer to "Special Handling" in this section.

*** - Refer to "83-Key Keyboard functions to Cordless Keyboard Mapping Chart."

**** - Uppercase for cursor keys can be selected by pressing left or right shift or entering the Numlock state (Alt + Fn + N).

***** - When Alt is pressed and the keyboard is in the Numlock state, the upper row of digits is used to enter ASCII codes for generating any character from the extended ASCII character set.

Cordless-Keyboard Character Codes (Part 1 of 4)

Key Number	Base Case	Upper Case	Ctrl	Alt	Fn
22	y	Y	EM (025)	*	-1
23	u	U	NAK (021)	*	-1
24	i	I	HT (009)	*	-1
25	o	O	SI (015)	*	-1
26	p	P	DLE (016)	*	**,*** (PrtScreen)
27	[	{	Esc (027)	(|) ***	-1
28	]	}	GS (029)	(~) ***	-1
29	CR	CR	LF (010)	-1	-1
30 Ctrl	-1	-1	-1	-1	-1
31	a	A	SOH (001)	*	-1
32	s	S	DC3 (019)	*	**,*** (Scroll Lock)
33	d	D	EOT (004)	*	-1
34	f	F	ACK (006)	*	-1
35	g	G	BELL (007)	*	-1
36	h	H	BS (008)	*	-1
37	j	J	LF (010)	*	-1
38	k	K	VT (011)	*	-1
39	l	L	FF (012)	*	-1
40	;	:	-1	-1	-1
41	’	”	-1	(‘) ***	1

* - Refer to “Extended Codes” in this section.

** - Refer to “Special Handling” in this section.

*** - Refer to “83-Key Keyboard functions to Cordless Keyboard Mapping Chart.”

**** - Uppercase for cursor keys can be selected by pressing left or right shift or entering the Numlock state (Alt + Fn + N).

***** - When Alt is pressed and the keyboard is in the Numlock state, the upper row of digits is used to enter ASCII codes for generating any character from the extended ASCII character set.

Cordless-Keyboard Character Codes (Part 2 of 4)

Key Number	Base Case	Upper Case	Ctrl	Alt	Fn	Alt + Ctrl
42	Cur.Up*	8 ****	-1	*	**,*** (Home)	
43 Left Shift	-1	-1	-1	-1	-1	
44	z	Z	SUB (026)	*	-1	
45	x	X	CAN (024)	*	-1	
46	c	C	EXT (003)	*	-1	
47	v	V	SYN (022)	*	-1	
48	b	B	STX (002)	*	**,*** (Break)	
49	n	N	SO (014)	*,***	***	
50	m	M	CR (013)	*	-1	
51	,	<	-1	-1	-1	
52	.	>	-1	(*) *	-1	
53	/	?	-1	\	-1	
54 Right Shift	-1	-1	-1	-1		
55	Cur.L *	4 ****	* Reverse Word	*	**,*** (PgUp)	**
56	Cur.R *	6 ****	* Advance Word	*	**,*** (PgDn)	** **

* - Refer to "Extended Codes" in this section.

** - Refer to "Special Handling" in this section.

*** - Refer to "83-Key Keyboard functions to Cordless Keyboard Mapping Chart."

**** - Uppercase for cursor keys can be selected by pressing left or right shift or entering the Numlock state (Alt + Fn + N).

***** - When Alt is pressed and the keyboard is in the Numlock state, the upper row of digits is used to enter ASCII codes for generating any character from the extended ASCII character set.

Cordless-Keyboard Character Codes (Part 3 of 4)

Key Number	Base Case	Upper Case	Ctrl	Alt	Fn	Alt + Ctrl
57 Alt	-1	-1	-1	-1	-1	
58	Space	Space	Space	Space	Space	
59 Caps Lock	-1	-1	-1	-1	-1	**
60	Ins.	0 ****	-1	*	-1	**
61	Del. *	. ****	-1	*	-1	**
62	Cur.Dn *	2 ****	-1	*	**,*** End	

* - Refer to "Extended Codes" in this section.

** - Refer to "Special Handling" in this section.

*** - Refer to "83-Key Keyboard functions to Cordless Keyboard Mapping Chart."

**** - Uppercase for cursor keys can be selected by pressing left or right shift or entering the Numlock state (Alt + Fn + N).

***** - When Alt is pressed and the keyboard is in the Numlock state, the upper row of digits is used to enter ASCII codes for generating any character from the extended ASCII character set.

Cordless-Keyboard Character Codes (Part 4 of 4)

Extended Codes

An extended code is used for certain functions that cannot be represented in the standard ASCII code. A character code of 000 (Nul) is returned in AL. This indicates that the system or application program should examine a second code that indicates the actual function. This code is returned in AH. This is the same for both the Cordless Keyboard and 83-key keyboard.

Second Code	Function
3	Null Character
15	⇤
16 through 25	Alt Q, W, E, R, T, Y, U, I, O, P
30 through 38	Alt A, S, D, F, G, H, J, K, L
44 through 50	Alt Z, X, C, V, B, N, M
59 through 68	Fn + 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 (Functions 1 through 10)
71	Home
72	Up Arrow
73	Page Up
75	← (Cursor Left)
77	→ (Cursor Right)
79	End
80	Down Arrow
81	Page Down
82	Ins (Insert)
83	Del (Delete)
84 through 93	F11 through F20 (Upper Case F1 through F10)
94 through 103	F21 through F30 (Ctrl F1 through F10)
104 through 113	F31 through F40 (Alt F1 through F10)
114	Fn/E or Ctrl/Fn/P (Start/Stop Echo to Printer)
115	Ctrl ← (Reverse Word)
116	Ctrl → (Advance Word)
117	Ctrl/End [Erase End of Line (EOL)]
118	Ctrl/PgDn [Erase to End of Screen (EOS)]
119	Ctrl/Home (Clear Screen and Home)
120 through 131	Alt/1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -, = (Keys 2 through 13)
132	Ctrl/PgUp (Top 25 Lines of Text and Home Cur.)
133 through 149	Reserved
150 through 190	Reserved for Non-Keyboard Scan Codes

Cordless Keyboard Extended Functions

Shift States

Most shift states are handled within the KEYBOARD routine, transparent to the system or application program. The current set of active shift states is available by 'calling' an entry point in the ROM KEYBOARD-routine. The following keys result in altered shift-states:

Shift

This key temporarily shifts keys 2 thru 13, 16 thru 28, 31 thru 41, and 44 thru 53 to upper case (base case if in Caps Lock state). The **Shift key** temporarily reverses the 'Num Lock' or 'non-Num-Lock' state of keys 42, 55, 56, and 60 thru 62.

Ctrl

This key temporarily shifts keys 3, 7, 12, 14, 16 thru 28, 30 thru 38, 42, 44 thru 50, 55, and 56 to the Ctrl state. The **Ctrl** key is used with the **Alt** and **Del** keys to cause the 'System Reset' function, with the Scroll Lock key to cause the 'Break' function, with the **Num Lock key** to cause the 'Pause' function, with the **Alt** and **Cursor Left** or **Right** for 'screen adjustment', with **Alt** and **Ins** to 'activate diagnostics', and with **Alt** and **CapsLock** to 'activate keyboard clicking'. These functions are described in "Special Handling" on the following pages.

Alt

The **Alt** key temporarily shifts keys 2 thru 13, 17 thru 26, 31 thru 39, and 44 thru 50 to the 'Alternate state'. The **Alt** key is used with the **Ctrl** and **Del** keys to cause the 'System Reset' function described in "Special Handling" on the following pages. The **Alt** key is also used with keys 27, 28, 41, and 53 to produce the characters under the key.

The **Alt** key has another use. This key allows the user to enter any character code from 0 to 255 into the system from the keyboard. The user must first put the keyboard in the 'Num Lock' state (concurrently press, first **Alt** then **Fn + n**). Then while holding down the **Alt** key type the decimal value of the character desired using keys 2 thru 11. The **Alt** key is then released. If more than three digits are typed, a modulo-256 result is created. These three digits are interpreted as a character code and are transmitted through the KEYBOARD routine to the system or application program. **Alt** is handled internal to the KEYBOARD routine.

Caps Lock

This key shifts keys 17 thru 25, 31 thru 39, and 44 thru 50 to 'upper case'. A second press of the **Caps Lock** key reverses the action. **Caps Lock** is handled internal to the KEYBOARD routine.

Shift-Key Priorities and Combinations

The following keys are listed in descending priority for translation in Interrupt Hex 48 and Interrupt hex 9 respectively:

1. Interrupt Hex 48.
   a. **Alt** key
   b. **Ctrl** key
   c. **Shift** key
2. Interrupt Hex 9
   a. **Ctrl**
   b. **Alt**
   c. **Shift**

Of the three keys listed, only **Alt** and **Ctrl** are a valid combination. If any other combination of the three keys is used, only the key with the higher priority is recognized by the system.

Special Handling

System Reset

The combination of the **Alt, Ctrl,** and **Del** keys causes the KEYBOARD routine to initiate the equivalent of a 'System Reset'.

Break

The combination of the **Fn** and **B** keys results in the KEYBOARD routine signaling Interrupt Hex 1A. The extended characters (AL = hex 00, AH = hex 00) are returned.

Pause

The combination of the **Fn** and **Q** keys causes the KEYBOARD-interrupt routine to loop, waiting for any key to be pressed. This provides a system or application-transparent method of temporarily suspending an operation such as list or print and then resuming the operation by pressing any other key. The key pressed to exit the 'Pause' mode is unused otherwise.

Print Screen

The combination of the **Fn** and **P** keys results in an interrupt, invoking the PRINT SCREEN routine. This routine works in the alphanumeric or graphics mode, with unrecognizable characters printing as blanks.

Scroll Lock

The combination of the **Fn** and **S** key is interpreted by appropriate application programs to indicate that the cursor-control keys should cause 'windowing' over the text rather than cursor movement. Pressing the 'Scroll Lock' combination a second time reverses the action. The KEYBOARD routine simply records the current shift state of 'Scroll Lock'. It is the responsibility of the system or application program to perform the function.

Functions 1 thru 10

The combination of the **Fn** key (15) and one of keys 2 thru 11 results in the corresponding 'Function' with key 2 being 'F1' up to key 11 being 'F10'.

Function Lock

Concurrently pressing first the **Fn** key and **Shift** key, and then pressing the **Esc** key causes keys 2 thru 11 to shift to their 'Function' states and remain there until the same combination is pressed again.

Screen Adjustment

The combination of the **Alt** key, **Ctrl** key, and either the **Left** or **Right** cursor movement key causes the screen to shift one character in the corresponding direction, up to a maximum of four.

Enable/Disable Keyboard Click

The combination of the **Alt**, **Ctrl**, and **Caps Lock** keys causes the keyboard audio feedback (click) to shift between 'on' and 'off'. The Power-On default is 'off'.

Run Diagnostics

The combination of the **Alt**, **Ctrl**, and **Ins keys** causes the system diagnostics stored in ROM to be initiated.

Phantom-Key Scan-Code (Hex 55)

The Phantom-Key scan-code is generated by the keyboard when an invalid combination of three or more keys is pressed. The keys pressed that caused the Phantom-Key scan-code are not put into the keyboard buffer, and are ignored by the keyboard microprocessor. The Phantom-Key scan-code is transmitted to BIOS where it is ignored.

Other Characteristics

The keyboard buffer is large enough to support a fast typist. If a key is pressed when the buffer is full, the character generated is ignored and the 'bell' is sounded. A larger buffer can be specified by modifying words at labels 'Buffer-Start' (hex 480) and 'Buffer-End' (hex 482) to point to another offset within segment hex 40.

The KEYBOARD routine suppresses the typematic action of the following keys: **Ctrl, Shift, Alt, Caps Lock, Insert**, and **Function.**

Function	Key Combinations	Description
System Reset	Alt + Ctrl + Del	Unconditional system reset
Break	Fn + B	Breaks program execution
Pause	Fn + Q	Resumable pause in program execution
Print Screen	Fn + P	
Function Lock	Fn and Shift then Esc (Held) concurrently)	Locks the number keys as Function keys (F1-F10) and B, Q, P, E, S, and the cursor control keys to their function states
Screen Adjustment	Alt + Ctrl + cursor right or cursor left	Allows the user to adjust the display's image left or right
Keyboard Click	Alt + Ctrl + CapsLock	Enables or disables the keyboard audio feedback click
Run Diagnostics	Alt + Ctrl + Ins	Initiates system ROM diagnostics
Keyboard Adventure Game	Esc	If the first key pressed after the system comes up in Cassette BASIC is Esc (key #1) then the Keyboard Adventure Game will be activated.
Cassette Autoload	Ctrl + Esc	If this is the first key sequence after the system comes up in Cassette BASIC then the screen will display 'Load "CAS1:",R followed by a Carriage Return. This allows a cassette program to be automatically loaded.

Cordless Keyboard Special Handling

Keyboard Usage

“Keyboard Usage” is a set of guidelines of key-usage when performing commonly-used functions.

Function	Keys	Comment
Home Cursor	Fn Home	Editors; word processors
Return to outermost menu	Fn Home	Menu driven applications
Move cursor up	Up Arrow	Full screen editor, word processor
Page up, scroll backwards 25 lines	Fn PgUp	Editors; word processors
Move cursor left	←	Text, command entry
Move cursor right	→	Text, command entry
Scroll to end of text place cursor at end of line	Fn End	Editors; word processors
Move cursor down	Down Arrow	Full screen editor, word processor
Page down, scroll forwards 25 lines and home	Fn PgDn	Editors; word processors
Start/Stop insert text at cursor, shift text right in buffer	Ins	Text, command entry

Keyboard - Commonly Used Functions (Part 1 of 3)

Function	Keys	Comment
Delete character at cursor	Del	Text, command entry
Destructive backspace	← Key 14	Text, command entry
Tab forward	⇥	Text entry
Tab reverse	⇤	Text entry
Clear screen and home	Ctrl Fn Home	
Scroll up	Up Arrow	In scroll lock mode
Scroll down	Down Arrow	In scroll lock mode
Scroll left	←	In scroll lock mode
Scroll right	→	In scroll lock mode
Delete from cursor to EOL (end of line)	Ctrl Fn End	Text, command entry
Exit/Escape	Esc	Editor, 1 level of menu and so on
Start/Stop Echo screen to printer	Fn PrtSc	Any time
Delete from cursor to EOS (end of screen)	Ctrl Fn PgDn	Text, command entry
Advance word	Ctrl →	Text entry
Reverse word	Ctrl ←	Text entry
Window Right	Ctrl →	When text is too wide to fit the screen

Keyboard - Commonly Used Functions (Part 2 of 3)

Function	Keys	Comment
Window Left	Ctrl ←	When text is too wide to fit the screen
Enter insert mode	Ins	Line Editor
Exit insert mode	Ins	Line Editor
Cancel current line	Esc	Command entry, text entry
Suspend system (Pause)	Ctrl Fn Pause	Stop list, stop program, and so on. Resumes on any key.
Break interrupt	Fn Break	Interrupt current process
System reset	Alt Ctrl Del	Reboot
Top of document and home cursor	Ctrl Fn PgUp	Editors, word processors
Standard function keys	Shift Fn/F1 through Fn/F10	Primary function keys
Secondary function keys	Shift F1-F10 Ctrl F1-F10 Alt F1-F10	Extra function keys if 10 are not sufficient.
Extra function keys	Alt keys 2 through 13 (1 through 9, 0) (–, =)	Line Editor
Extra function keys	Alt A through Z	Used when function starts with the same letter as one of the alpha keys.

Keyboard - Commonly Used Functions (Part 3 of 3)

Function	Key
Carriage return	↵ (Enter)
Line feed	Ctrl ↵ (Enter)
Bell	Ctrl G
Home	Fn Home
Cursor up	Up Arrow
Cursor down	Down Arrow
Cursor left	←
Cursor right	→
Advance one word	Ctrl ←
Reverse one word	Ctrl →
Insert	Ins
Delete	Del
Clear screen	Ctrl Fn Home
Freeze output	Fn Pause
Tab advance	⇥
Stop Execution (break)	Fn Break
Delete current line	Esc
Delete to end of line	Ctrl Fn End
Position cursor to end of line	Fn End

BASIC Screen Editor Special Functions

Function	Key
Suspend	Fn Pause
Echo to printer	Fn Echo
Stop echo to printer	Fn Echo
Exit current function (break)	Fn Break
Backspace	◄── Key 14
Line feed	Ctrl ↵ (Enter)
Cancel line	Esc
Copy character	Fn F1 or ──►
Copy until match	Fn F2
Copy remaining	Fn F3
Skip character	Del
Skip until match	Fn F4
Enter insert mode	Ins
Exit insert mode	Ins
Make new line the template	Fn F5
String separator in REPLACE	Fn F6
End of file in keyboard input	Fn F6

DOS Special Functions

Non-Keyboard Scan-code Architecture

The architecture of the IBM PC*jr* BIOS is designed to also receive scan codes above those generated by the keyboard to accommodate any future device.

The keyboard generates scan codes from hex 1 to 55 and FF. Any scan codes above hex 55 (56 thru 7E for 'make' codes and D6 thru FE for 'break' codes) are processed by BIOS in the following manner:

1. If the incoming 'make' scan code falls within the range of the translate table, whose address is pointed to by BIOS Interrupt Hex 49, it is translated into the corresponding scan code. Any incoming 'break' codes above hex D5 are ignored.

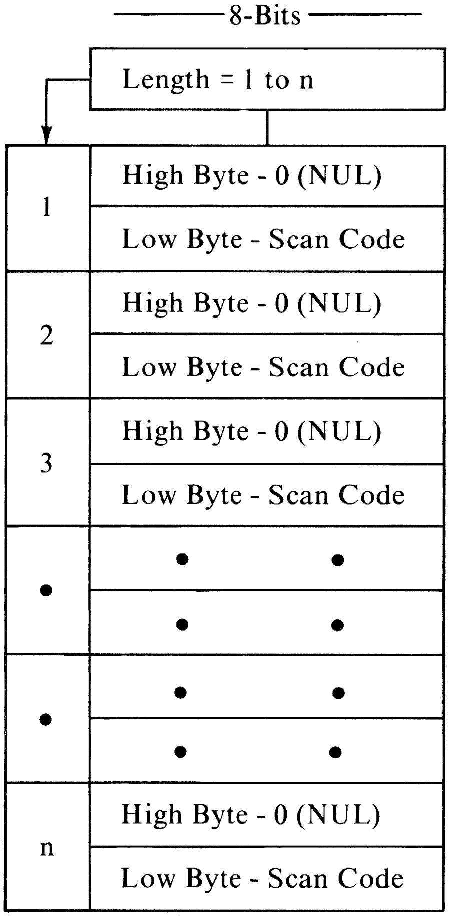
2. If the new translated scan code is less than hex 56, it is processed by BIOS as a keyboard scan-code and the same data is placed in the BIOS keyboard buffer.
3. If the translated scan-code is greater than hex 55 or the incoming scan-code is outside the range of the translate table, hex 40 is added, creating a new extended-scan-code. The new extended-scan-code is then placed in the BIOS keyboard buffer with the character code of 00(null). This utilizes the range hex 96 thru BE for scan codes hex 56 thru 7E respectively.

The default translate-table maps scan codes hex 56 thru 6A to existing keyboard-values. Scan codes hex 6B thru BE are mapped (by adding hex 40) to extended codes of hex AB thru FE, since these are out side the range of the default translate-table.

Users can modify Interrupt Hex 49 to address their own translate table if mapping differences are desired.

The translate table format is:

	Description
0	Length - The number of non-keyboard scan-codes that are mapped within the table (from 1 to n).
1 to n	Word with low-order byte representing the scan-code-mapped values relative to the input values in the range of hex 56 thru 7E.



Translate Table Format

With this architecture, all keyboard scan-codes can be intercepted thru Interrupt Hex 9 and all non-keyboard scan-codes can be intercepted thru Interrupt Hex 48.

The following is a chart showing the default values of the translate table in BIOS.

Length = 20 mapped values		
Input Scan Code	**Mapped Value**	**Keyboard Character**
86	72	(cursor up)
87	73	PgUp
88	77	(cursor right)
89	81	PgDn
90	80	(cursor down)
91	79	End
92	75	(cursor left)
93	71	Home
94	57	Space
95	28	Enter
96	17	W
97	18	E
98	31	S
99	45	X
100	44	Z
101	43	\
102	30	A
103	16	Q
104	15	Tab
105	1	Esc

Translate Table Default Values

Scan Codes (Hex)	Type of Scan Code
1 - 55	Normal Keyboard Scan Code (Make)
56 - 7E	Non-Keyboard Scan Code (Make)
81 - D5	Normal Keyboard Scan Code (Break)
D6 - FE	Non-Keyboard Scan Code (Break)
FF	Keyboard Buffer Full

Scan-Code Map

Notes:

BIOS Cassette Logic

Software Algorithms - Interrupt Hex 15

The CASSETTE routine is called by the request type in AH. The address of the bytes to be 'read' from or 'written' to the tape is specified by DS:BX and the number of bytes to be 'read' or 'written' is specified by CX. The actual number of bytes 'read' is returned in DX. The read block and write block automatically turn the cassette motor on at the start and off at the end. The request types in AH and the cassette status descriptions follow:

Request Type	Function
AH = 0	Turn Cassette Motor On
AH = 1	Turn Cassette Motor Off
AH = 2	Read Tape Block Read CX bytes into memory starting at Address DS:BX Return actual number of bytes read in DX Return Cassette Status in AH
AH = 3	Write Tape Block Write CX bytes onto cassette starting at Address DS:BX Return Cassette Status in AH

AH Request Types

Cassette Status	Description
AH = 00	No Errors
AH = 01	Cyclic Redundancy Check (CRC) Error in Read Block
AH = 02	No Data Transitions
AH = 04	No Leader
AH = 80	Invalid Command
Note: The carry flag will be set on any error.	

AH Cassette Status

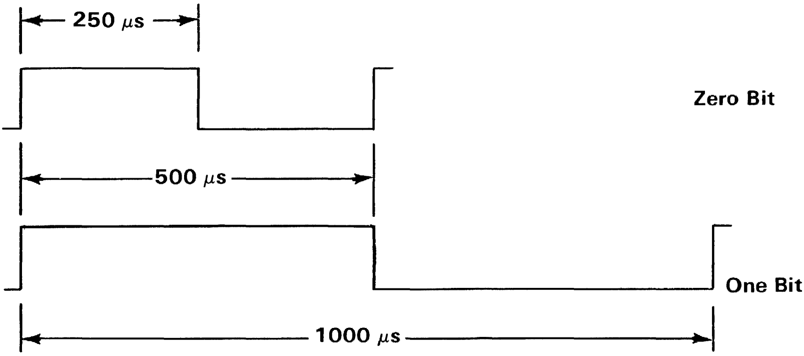
Cassette Write

The WRITE-BLOCK routine 'writes' a tape block onto the cassette tape. The tape block is described in "Data Record Architecture" later in this section.

The WRITE-BLOCK routine 'turns on' the cassette drive motor and 'writes' the leader (256 bytes of all 1's) to the tape, 'writes' a synchronization bit (0), and then 'writes' a synchronization byte (ASCII character hex 16). Next, the routine 'writes' the number of data bytes specified by CX. After each data block of 256 bytes, a 2-byte cyclic redundancy check (CRC) is 'written'. The data bytes are taken from the memory location 'pointed' at by DS:BX.

The WRITE-BLOCK routine 'disassembles' and 'writes' the byte a bit-at-a-time to the cassette. The method used is to 'set' Timer 2 to the period of the desired data bit. The timer is 'set' to a period of 1.0 millisecond for a 1 bit and 0.5 millisecond for a 0 bit.

The timer is 'set' to mode 3, which means the timer outputs a square wave with a period given by its count register. The timer's period is changed on the fly for each data byte 'written' to the cassette. If the number of data bytes to be 'written' is not an integral multiple of 256, then, after the last desired data byte from memory has been 'written', the data block is extended to 256 bytes of writing multiples of the last data byte. The last block is closed with two CRC bytes as usual. After the last data-block, a trailer consisting of four bytes of all 1 bits is 'written'. Finally, the cassette motor is 'turned off', if there are no errors reported by the routine. All 8259 interrupts are 'disabled' during cassette-write operations.



Cassette-Write Timing Chart

Cassette Read

The READ-BLOCK routine 'turns on' the cassette drive motor and then delays for approximately 0.5 second to allow the motor to come up to speed.

The READ-BLOCK routine then searches for the leader and must detect all 1 bits for approximately 1/4 of the leader length before it can look for the sync (0) bit. After the sync bit is detected, the sync byte

(ASCII character hex 16) is 'read'. If the sync byte is 'read' correctly, the data portion can be 'read'. If a correct sync byte is not found, the routine goes back and searches for the leader again. The data is 'read' a bit-at-a-time and 'assembled' into bytes. After each byte is 'assembled', it is 'written' into memory at location DS:BX and BX is incremented by 1.

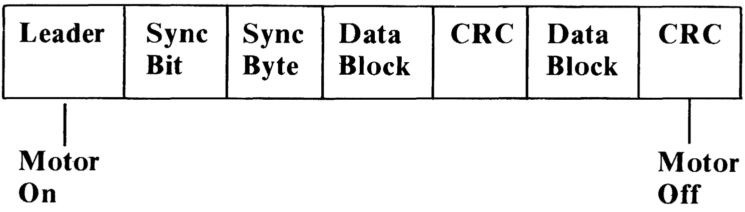
After each multiple of 256 data bytes is 'read', the CRC is 'read' and 'compared' to the CRC generated. If a CRC error is detected, the routine exits with the carry flag 'set' to indicate an error and the status of AH 'set' to hex 01. DX contains the number of bytes 'written' into memory.

All 8259 interrupts are 'disabled' during the cassette-'read' operations.

Data Record Architecture

The WRITE-BLOCK routine uses the following format to record a tape block onto a cassette tape:

(CASSETTE TAPE BLOCK)



Cassette Write-Block Format

Component	Description
Leader	256 Bytes (of All 1's)
Sync Bit	One 0 bit
Sync Byte	ASCII Character hex 16
Data Blocks	256 Bytes in Length
CRC	2 Bytes for each Data Block

Data Record Components

Error Detection

Error detection is handled through software. A CRC is used to detect errors. The polynomial used is G(X) = $X^{16} + X^{12} + X^5 + 1$, which is the polynomial used by the synchronous data link control interface. Essentially, as bits are 'written' to or 'read' from the cassette tape they are passed through the CRC register in software. After a block of data is 'written', the complemented value of the calculated CRC register is 'written' on the tape. Upon reading the cassette data, the CRC bytes are 'read' and 'compared' to the generated CRC value. If the read CRC does not equal the generated CRC, the processor's carry flag is 'set' and the status of AH is 'set' to hex 01, which indicates a CRC error has occurred. Also, the routine is exited on a CRC error.

Notes:

Appendixes

Contents

```
                                ;
                                ;-----------------------------------------------------------
                                ; <CAVEAT EMPTOR>:
                                ;
                                ;       THE  BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH
                                ;       SOFTWARE INTERRUPTS ONLY.  ANY ADDRESSES PRESENT IN
                                ;       THE LISTINGS  ARE INCLUDED  ONLY FOR  COMPLETENESS,
                                ;       NOT FOR  REFERENCE.   APPLICATIONS WHICH  REFERENCE
                                ;       ABSOLUTE  ADDRESSES  WITHIN THIS  CODE  VIOLATE THE
                                ;       STRUCTURE AND DESIGN OF BIOS.
                                ;-----------------------------------------------------------
                                ;-------------------------------------
                                ;               EQUATES
                                ;-------------------------------------
 = 0060                         PORT_A          EQU     60H      ; 8255 PORT A ADDR
 = 0038                         CPUREG          EQU     38H      ; MASK FOR CPU REG BITS
 = 0007                         CRTREG          EQU     7        ; MASK FOR CRT REG BITS
 = 0061                         PORT_B          EQU     61H      ; 8255 PORT B ADDR
 = 0062                         PORT_C          EQU     62H      ; 8255 PORT C ADDR
 = 0063                         CMD_PORT        EQU     63H
 = 0089                         MODE_8255       EQU     10001001B
 = 0020                         INTA00          EQU     20H      ; 8259 PORT
 = 0021                         INTA01          EQU     21H      ; 8259 PORT
 = 0020                         EOI             EQU     20H
 = 0040                         TIMER           EQU     40H
 = 0043                         TIM_CTL         EQU     43H      ; 8253 TIMER CONTROL PORT ADDR
 = 0040                         TIMER0          EQU     40H      ; 8253 TIMER/CNTER 0 PORT ADDR
 = 0061                         KB_CTL          EQU     61H      ; CONTROL BITS FOR KEYBOARD
 = 03DA                         VGA_CTL         EQU     3DAH     ; VIDEO GATE ARRAY CONTROL PORT
 = 00A0                         NMI_PORT        EQU     0A0H     ; NMI CONTROL PORT
 = 00B0                         PORT_B0         EQU     0B0H
 = 03DF                         PAGREG          EQU     03DFH    ; CRT/CPU PAGE REGISTER
 = 0060                         KBPORT          EQU     060H     ; KEYBOARD PORT
 = 4000                         DIAG_TABLE_PTR  EQU     4000H
 = 2000                         MINI            EQU     2000H
                                ;-------------------------------------
                                ;         DISKETTE EQUATES
                                ;-------------------------------------
 = 00F2                         NEC_CTL         EQU     0F2H     ; CONTROL PORT FOR THE DISKETTE
 = 0080                         FDC_RESET       EQU     80H      ; RESETS THE NEC (FLOPPY DISK
                                                                 ; CONTROLLER).  0 RESETS,
                                                                 ; 1 RELEASES THE RESET
 = 0020                         WD_ENABLE       EQU     20H      ; ENABLES WATCH DOG TIMER IN NEC
 = 0040                         WD_STROBE       EQU     40H      ; STROBES WATCHDOG TIMER
 = 0001                         DRIVE_ENABLE    EQU     01H      ; SELECTS AND ENABLES DRIVE

 = 00F4                         NEC_STAT        EQU     0F4H     ; STATUS REGISTER FOR THE NEC
 = 0020                         BUSY_BIT        EQU     20H      ; BIT = 0 AT END OF EXECUTION PHASE
 = 0040                         DIO             EQU     40H      ; INDICATES DIRECTION OF TRANSFER
 = 0080                         RQM             EQU     80H      ; REQUEST FOR MASTER
 = 00F5                         NEC_DATA        EQU     0F5H     ; DATA PORT FOR THE NEC
                                ;-------------------------------------
                                ;        8088 INTERRUPT LOCATIONS
                                ;-------------------------------------
 0000                           ABS0    SEGMENT AT 0
 0008                                   ORG     2*4
 0008                           NMI_PTR         LABEL    WORD
 000C                                   ORG     3*4
 000C                           INT3_PTR        LABEL   WORD
 0014                                   ORG     5*4
 0014                           INT5_PTR        LABEL   WORD
 0020                                   ORG     8*4
 0020                           INT_PTR         LABEL   DWORD
 0040                                   ORG     10H*4
 0040                           VIDEO_INT       LABEL   WORD
 0070                                   ORG     1CH*4
 0070                           INT1C_PTR       LABEL   WORD
 0074                                   ORG     1DH*4
 0074                           PARM_PTR        LABEL   DWORD    ; POINTER TO VIDEO PARMS
 0060                                   ORG     18H*4
 0060                           BASIC_PTR       LABEL   WORD     ; ENTRY POINT FOR CASSETTE BASIC
 0078                                   ORG     01EH*4           ; INTERRUPT 1EH
 0078                           DISK_POINTER    LABEL   DWORD
 007C                                   ORG     01FH*4           ; LOCATION OF POINTER
 007C                           EXT_PTR LABEL   DWORD            ; POINTER TO EXTENSION
 0110                                   ORG     044H*4
 0110                           CSET_PTR        LABEL   DWORD    ; POINTER TO DOT PATTERNS
 0120                                   ORG     048H*4
 0120                           KEY62_PTR       LABEL   WORD     ; POINTER TO 62 KEY KEYBOARD CODE
 0124                                   ORG     049H*4
 0124                           EXST            LABEL   WORD     ; POINTER TO EXT. SCAN TABLE
 0204                                   ORG     081H*4
 0204                           INT81   LABEL   WORD
 0208                                   ORG     082H*4
 0208                           INT82   LABEL   WORD
 0224                                   ORG     089H*4
 0224                           INT89   LABEL   WORD
 0400                                   ORG     400H
 0400                           DATA_AREA       LABEL   BYTE     ; ABSOLUTE LOCATION OF DATA SEGMENT
 0400                           DATA_WORD       LABEL   WORD
 7C00                                   ORG     7C00H
 7C00                           BOOT_LOCN       LABEL   FAR
 7C00                           ABS0    ENDS
```

```
                               ;------------------------------------------------
                               ; STACK -- USED DURING INITIALIZATION ONLY
                               ;------------------------------------------------
0000                           STACK   SEGMENT AT 30H
0000     80 [                          DW      128 DUP(?)
               ????
                     ]

0100                           TOS     LABEL   WORD
0100                           STACK   ENDS
                               ;----------------------------------------
                               ;       ROM BIOS DATA AREAS
                               ;----------------------------------------
0000                           DATA    SEGMENT AT 40H
0000     04 [                  RS232_BASE      DW      4 DUP(?) ; ADDRESSES OF RS232 ADAPTERS
               ????
                     ]

0008     04 [                  PRINTER_BASE    DW      4 DUP(?) ; ADDRESSES OF PRINTERS
               ????
                     ]

0010  ????                     EQUIP_FLAG      DW      ?       ; INSTALLED HARDWARE
0012  ??                       KBD_ERR         DB      ?       ; COUNT OF KEYBOARD TRANSMIT ERRORS
0013  ????                     MEMORY_SIZE     DW      ?       ; USABLE MEMORY SIZE IN K BYTES
0015  ????                     TRUE_MEM        DW      ?       ; REAL MEMORY SIZE IN K BYTES
                               ;----------------------------------------
                               ;       KEYBOARD DATA AREAS
                               ;----------------------------------------
0017  ??                       KB_FLAG         DB      ?
                               ;----- SHIFT FLAG EQUATES WITHIN KB_FLAG
= 0040                         CAPS_STATE      EQU     40H     ; CAPS LOCK STATE HAS BEEN TOGGLED
= 0020                         NUM_STATE       EQU     20H     ; NUM LOCK STATE HAS BEEN TOGGLED
= 0008                         ALT_SHIFT       EQU     08H     ; ALTERNATE SHIFT KEY DEPRESSED
= 0004                         CTL_SHIFT       EQU     04H     ; CONTROL SHIFT KEY DEPRESSED
= 0002                         LEFT_SHIFT      EQU     02H     ; LEFT SHIFT KEY DEPRESSED
= 0001                         RIGHT_SHIFT     EQU     01H     ; RIGHT SHIFT KEY DEPRESSED
0018  ??                       KB_FLAG_1       DB      ?       ; SECOND BYTE OF KEYBOARD STATUS
= 0080                         INS_SHIFT       EQU     80H     ; INSERT KEY IS DEPRESSED
= 0040                         CAPS_SHIFT      EQU     40H     ; CAPS LOCK KEY IS DEPRESSED
= 0020                         NUM_SHIFT       EQU     20H     ; NUM LOCK KEY IS DEPRESSED
= 0010                         SCROLL_SHIFT    EQU     10H     ; SCROLL LOCK KEY IS DEPRESSED
= 0008                         HOLD_STATE      EQU     08H     ; SUSPEND KEY HAS BEEN TOGGLED
= 0004                         CLICK_ON        EQU     04H     ; INDICATES THAT AUDIO FEEDBACK IS
                                                               ; ENABLED
= 0002                         CLICK_SEQUENCE  EQU     02H     ; OCURRNCE OF ALT-CTRL-CAPSLOCK HAS
                                                               ; OCCURED
0019  ??                       ALT_INPUT       DB      ?       ; STORAGE FOR ALTERNATE KEYPAD
                                                               ; ENTRY
001A  ????                     BUFFER_HEAD     DW      ?       ; POINTER TO HEAD OF KEYBOARD BUFF
001C  ????                     BUFFER_TAIL     DW      ?       ; POINTER TO TAIL OF KEYBOARD BUFF
001E     10 [                  KB_BUFFER       DW      16 DUP(?) ; ROOM FOR 15 ENTRIES
               ????
                     ]

                               ;------ HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
= 0045                         NUM_KEY         EQU     69      ; SCAN CODE FOR NUMBER LOCK
= 0046                         SCROLL_KEY      EQU     70      ; SCROLL LOCK KEY
= 0038                         ALT_KEY         EQU     56      ; ALTERNATE SHIFT KEY SCAN CODE
= 001D                         CTL_KEY         EQU     29      ; SCAN CODE FOR CONTROL KEY
= 003A                         CAPS_KEY        EQU     58      ; SCAN CODE FOR SHIFT LOCK
= 002A                         LEFT_KEY        EQU     42      ; SCAN CODE FOR LEFT SHIFT
= 0036                         RIGHT_KEY       EQU     54      ; SCAN CODE FOR RIGHT SHIFT
= 0052                         INS_KEY         EQU     82      ; SCAN CODE FOR INSERT KEY
= 0053                         DEL_KEY         EQU     83      ; SCAN CODE FOR DELETE KEY
                               ;----------------------------------------
                               ;       DISKETTE DATA AREAS
                               ;----------------------------------------
003E  ??                       SEEK_STATUS     DB      ?       ; DRIVE RECALIBRATION STATUS
                                                               ; BIT 0 = DRIVE NEEDS RECAL BEFORE
                                                               ;   NEXT SEEK IF BIT IS = 0
003F  ??                       MOTOR_STATUS    DB      ?       ; MOTOR STATUS
                                                               ; BIT 0 = DRIVE 0 IS CURRENTLY
                                                               ;   RUNNING
0040  ??                       MOTOR_COUNT     DB      ?       ; TIME OUT COUNTER FOR DRIVE
                                                               ; TURN OFF
= 0025                         MOTOR_WAIT      EQU     37      ; 2 SECS OF COUNTS FOR MOTOR
                                                               ; TURN OFF
0041  ??                       DISKETTE_STATUS DB      ?       ; RETURN CODE STATUS BYTE
= 0080                         TIME_OUT        EQU     80H     ; ATTACHMENT FAILED TO RESPOND
= 0040                         BAD_SEEK        EQU     40H     ; SEEK OPERATION FAILED
= 0020                         BAD_NEC         EQU     20H     ; NEC CONTROLLER HAS FAILED
= 0010                         BAD_CRC         EQU     10H     ; BAD CRC ON DISKETTE READ
= 0009                         DMA_BOUNDARY    EQU     09H     ; ATTEMPT TO DMA ACROSS 64K
                                                               ; BOUNDARY
= 0008                         BAD_DMA         EQU     08H     ; DMA OVERRUN ON OPERATION
= 0004                         RECORD_NOT_FND  EQU     04H     ; REQUESTED SECTOR NOT FOUND
= 0003                         WRITE_PROTECT   EQU     03H     ; WRITE ATTEMPTED ON WRITE
                                                               ; PROTECTED DISK
= 0002                         BAD_ADDR_MARK   EQU     02H     ; ADDRESS MARK NOT FOUND
= 0001                         BAD_CMD         EQU     01H     ; BAD COMMAND GIVEN TO DISKETTE I/O
0042     07 [                  NEC_STATUS      DB      7 DUP(?) ; STATUS BYTES FROM NEC
               ??
                   ]

= 0020                         SEEK_END        EQU     20H
= 012C                         THRESHOLD       EQU     300     ; NUMBER OF TIMER-0 TICKS TILL
                                                               ; ENABLE
= 00AF                         PARM0           EQU     0AFH    ; PARAMETER 0 IN THE DISK_PARM
                                                               ; TABLE
= 0003                         PARM1           EQU     3       ; PARAMETER 1
= 0019                         PARM9           EQU     25      ; PARAMETER 9
= 0004                         PARM10          EQU     4       ; PARAMETER 10
```

```
                                ;----------------------------------------
                                ;       VIDEO DISPLAY DATA AREA
                                ;----------------------------------------
0049  ??                        CRT_MODE        DB      ?       ; CURRENT CRT MODE
004A  ????                      CRT_COLS        DW      ?       ; NUMBER OF COLUMNS ON SCREEN
004C  ????                      CRT_LEN         DW      ?       ; LENGTH OF REGEN IN BYTES
004E  ????                      CRT_START       DW      ?       ; STARTING ADDRESS IN REGEN BUFFER
0050     08 [                   CURSOR_POSN     DW      8 DUP(?) ; CURSOR FOR EACH OF UP TO 8 PAGES
              ????
                  ]

0060  ????                      CURSOR_MODE     DW      ?       ; CURRENT CURSOR MODE SETTING
0062  ??                        ACTIVE_PAGE     DB      ?       ; CURRENT PAGE BEING DISPLAYED
0063  ????                      ADDR_6845       DW      ?       ; BASE ADDRESS FOR ACTIVE DISPLAY
                                                                ; CARD
0065  ??                        CRT_MODE_SET    DB      ?       ; CURRENT SETTING OF THE
                                                                ; CRT MODE REGISTER
0066  ??                        CRT_PALLETTE    DB      ?       ; CURRENT PALETTE MASK SETTING
                                ;----------------------------------------
                                ;       CASSETTE DATA AREA
                                ;----------------------------------------
0067  ????                      EDGE_CNT        DW      ?       ; TIME COUNT AT DATA EDGE
0069  ????                      CRC_REG         DW      ?       ; CRC REGISTER
006B  ??                        LAST_VAL        DB      ?       ; LAST INPUT VALUE
                                ;----------------------------------------
                                ;       TIMER DATA AREA
                                ;----------------------------------------
006C  ????                      TIMER_LOW       DW      ?       ; LOW WORD OF TIMER COUNT
006E  ????                      TIMER_HIGH      DW      ?       ; HIGH WORD OF TIMER COUNT
0070  ??                        TIMER_OFL       DB      ?       ; TIMER HAS ROLLED OVER SINCE LAST
                                                                ; READ
                                ;----------------------------------------
                                ;       SYSTEM DATA AREA
                                ;----------------------------------------
0071  ??                        BIOS_BREAK      DB      ?       ; BIT 7=1 IF BREAK KEY HAS BEEN HIT
0072  ????                      RESET_FLAG      DW      ?       ; WORD=1234H IF KEYBOARD RESET
                                                                ; UNDERWAY
                                ;----------------------------------------
                                ;       EXTRA DISKETTE DATA AREAS
                                ;----------------------------------------
0074  ??                        TRACK0          DB      ?
0075  ??                        TRACK1          DB      ?
0076  ??                        TRACK2          DB      ?
0077  ??                                        DB      ?
                                ;------------------------------------------------
                                ;       PRINTER AND RS232 TIME-OUT VARIABLES
                                ;------------------------------------------------
0078     04 [                   PRINT_TIM_OUT   DB      4 DUP(?)
              ??
                ]

007C     04 [                   RS232_TIM_OUT   DB      4 DUP(?)
              ??
                ]


                                ;----------------------------------------
                                ;       ADDITIONAL KEYBOARD DATA AREA
                                ;----------------------------------------
0080  ????                      BUFFER_START    DW      ?
0082  ????                      BUFFER_END      DW      ?
0084  ??                        INTR_FLAG       DB      ?       ; FLAG TO INDICATE AN INTERRUPT
                                                                ; HAPPENED
                                ;----------------------------------------
                                ;  62 KEY KEYBOARD DATA AREA
                                ;----------------------------------------
0085  ??                        CUR_CHAR        DB      ?       ; CURRENT CHARACTER FOR TYPAMATIC
0086  ??                        VAR_DELAY       DB      ?       ; DETERMINES WHEN INITIAL DELAY IS
                                                                ; OVER
= 000F                          DELAY_RATE      EQU     0FH     ; INCREASES INITIAL DELAY
0087  ??                        CUR_FUNC        DB      ?       ; CURRENT FUNCTION
0088  ??                        KB_FLAG_2       DB      ?       ; 3RD BYTE OF KEYBOARD FLAGS
= 0004                          RANGE           EQU     4       ; NUMBER OF POSITIONS TO SHIFT
                                                                ; DISPLAY
                                ;----------------------------------------
                                ;  BIT ASSIGNMETS FOR KB_FLAG_2
                                ;----------------------------------------
= 0080                          FN_FLAG         EQU     80H
= 0040                          FN_BREAK        EQU     40H
= 0020                          FN_PENDING      EQU     20H
= 0010                          FN_LOCK         EQU     10H
= 0008                          TYPE_OFF        EQU     08H
= 0004                          HALF_RATE       EQU     04H
= 0002                          INIT_DELAY      EQU     02H
= 0001                          PUTCHAR         EQU     01H
0089  ??                        HORZ_POS        DB      ?       ; CURRENT VALUE OF HORIZONTAL
                                                                ; START PARM
008A  ??                        PAGDAT          DB      ?       ; IMAGE OF DATA WRITTEN TO PAGREG
008B                            DATA    ENDS
                                ;----------------------------------------
                                ;       EXTRA DATA AREA
                                ;----------------------------------------
0000                            XXDATA  SEGMENT AT 50H
0000  ??                        STATUS_BYTE     DB      ?
                                ; THE FOLLOWING AREA IS USED ONLY DURING DIAGNOSTICS
                                ; (POST AND ROM RESIDENT)
0001  ??                        DCP_MENU_PAGE   DB      ?       ; TO CURRENT PAGE FOR DIAG. MENU
0002  ????                      DCP_ROW_COL     DW      ?       ; CURRENT ROW/COLUMN COORDINATES
                                                                ; FOR DIAG MENU
0004  ??                        WRAP_FLAG       DB      ?       ; INTERNAL/EXTERNAL 8250 WRAP
                                                                ; INDICATOR
```

```
0005  ??                      MFG_TST         DB      ?       ; INITIALIZATION FLAG
0006  ????                    MEM_TOT         DW      ?       ; WORD EQUIV. TO HIGHEST SEGMENT IN
                                                              ; MEMORY
0008  ????                    MEM_DONES       DW      ?       ; CURRENT SEGMENT VALUE FOR
                                                              ; BACKGROUND MEM TEST
000A  ????                    MEM_DONEO       DW      ?       ; CURRENT OFFSET VALUE FOR
                                                              ; BACKGROUND MEM TEST
000C  ????                    INT1CO          DW      ?       ; SAVE AREA FOR INTERRUPT 1C
                                                              ; ROUTINE
000E  ????                    INT1CS          DW      ?
0010  ??                      MENU_UP         DB      ?       ; FLAG TO INDICATE WHETHER MENU IS
                                                              ; ON SCREEN (FF=YES, 0=NO)
0011  ??                      DONE128         DB      ?       ; COUNTER TO KEEP TRACK OF 128 BYTE
                                                              ; BLOCKS TESTED BY BGMEM
0012  ????                    KBDONE          DW      ?       ; TOTAL K OF MEMORY THAT HAS BEEN
                                                              ; TESTED BY BACKGROUND MEM TEST
                              ;----------------------------------------
                              ;       POST DATA AREA
                              ;----------------------------------------
0014  ????                    IO_ROM_INIT     DW      ?       ; POINTR TO OPTIONAL I/O ROM INIT
                                                              ; ROUTINE
0016  ????                    IO_ROM_SEG      DW      ?       ; POINTER TO IO ROM SEGMENT
0018  ??                      POST_ERR        DB      ?       ; FLAG TO INDICATE ERROR OCCURRED
                                                              ; DURING POST
0019      09 [                MODEM_BUFFER    DB      9 DUP(?) ; MODEM RESPONSE BUFFER
                ??
                    ]

                                                              ; (MAX 9 CHARS)
0022  ????                    MFG_RTN         DW      ?       ; POINTER TO MFG. OUTPUT ROUTINE
0024  ????                                    DW      ?
                              ;----------------------------------------
                              ;     SERIAL PRINTER DATA
                              ;----------------------------------------
0026  ????                    SP_FLAG         DW      ?
0028  ??                      SP_CHAR         DB      ?
                                                              ; THE FOLLOWING SIX ENTRIES ARE
                                                              ; DATA PERTAINING TO NEW STICK
0029  ????                    NEW_STICK_DATA  DW      ?       ; RIGHT STICK DELAY
002B  ????                                    DW      ?       ; RIGHT BUTTON A DELAY
002D  ????                                    DW      ?       ; RIGHT BUTTON B DELAY
002F  ????                                    DW      ?       ; LEFT STICK DELAY
0031  ????                                    DW      ?       ; LEFT BUTTON A DELAY
0033  ????                                    DW      ?       ; LEFT BUTTON B DELAY
0035  ????                                    DW      ?       ; RIGHT STICK LOCATION
0037  ????                                    DW      ?       ; UNUSED
0039  ????                                    DW      ?       ; UNUSED
003B  ????                                    DW      ?       ; LEFT STICK POSTITON
003D                          XXDATA  ENDS
                              ;----------------------------------------
                              ;     DISKETTE DATA AREA
                              ;----------------------------------------
0000                          DKDATA  SEGMENT AT 60H
0000  ??                      NUM_DRIVE       DB      ?
0001  ??                      DUAL            DB      ?
0002  ??                      OPERATION       DB      ?
0003  ??                      DRIVE           DB      ?
0004  ??                      TRACK           DB      ?
0005  ??                      HEAD            DB      ?
0006  ??                      SECTOR          DB      ?
0007  ??                      NUM_SECTOR      DB      ?
0008  ??                      SEC             DB      ?
                              ;    FORMAT ID
0009      08 [                TK_HD_SC        DB      8 DUP(0,0,0,0)  ;TRACK,HEAD,SECTOR,NUM OF
                00
                00
                00
                00
                    ]

                                                              ; SECTOR
                              ;    BUFFER FOR READ AND WRITE OPERATION
= 0200                        DK_BUF_LEN      EQU     512     ; 512 BYTES/SECTOR
0029  0200 [                  READ_BUF        DB      DK_BUF_LEN DUP(0)
                00
                    ]

0229  0100 [                  WRITE_BUF       DB      (DK_BUF_LEN/2) DUP(6DH,0BH)
                6D
                0B
                    ]

                              ;    INFO FLAGS
0429  ??                      REQUEST_IN      DB      ?      ;SELECTION CHARACTER
042A  ??                      DK_EXISTED      DB      ?
042B  ??                      DK_FLAG         DB      ?
042C  ????                    RAN_NUM         DW      ?
042E  ????                    SEED            DW      ?
                              ;    SPEED TEST VARIABLES
0430  ????                    DK_SPEED        DW      ?
0432  ????                    TIM_1           DW      ?
0434  ????                    TIM_L_1         DW      ?
0436  ????                    TIM_2           DW      ?
0438  ????                    TIM_L_2         DW      ?
043A  ????                    FRACT_H         DW      ?
043C  ????                    FRACT_L         DW      ?
043E  ????                    PART_CYCLE      DW      ?
0440  ????                    WHOLE_CYCLE     DW      ?
0442  ????                    HALF_CYCLE      DW      ?
```

```
                               ;        ERROR PARAMETERS
0444  ??                       DK_ER_OCCURED    DB      ?       ;ERROR HAS OCCURRED
0445  ??                       DK_ER_L1         DB      ?       ;CUSTOMER ERROR LEVEL
0446  ??                       DK_ER_L2         DB      ?       ;SERVICE ERROR LEVEL
0447  ??                       ER_STATUS_BYTE   DB      ?       ;STATUS BYTE RETURN FROM INT 13H
                               ;        LANGUAGE TABLE
0448  ??                       LANG_BYTE        DB      ?       ; PORT B0 TO DETERMINE WHICH
                                                                ; LANGAGE TO USE
0449                           DKDATA  ENDS
                               ;----------------------------------------
                               ;        VIDEO DISPLAY BUFFER
                               ;----------------------------------------
0000                           VIDEO_RAM        SEGMENT AT 0B800H
0000  4000 [                           DB       16384 DUP(?)
               ??
                   ]

4000                           VIDEO_RAM        ENDS
                               ;-----------------------------------------
                               ;        ROM RESIDENT CODE                :
                               ;-----------------------------------------
0000                           CODE     SEGMENT PAGE
                                        ASSUME  CS:CODE,DS:ABS0,ES:NOTHING,SS:STACK

0000  31 35 30 34 30 33                 DB      '1504036 COPR. IBM 1981,1983'   ; COPYRIGHT NOTICE
      36 20 43 4F 50 52
      2E 20 49 42 4D 20
      31 39 38 31 2C 31
      39 38 33

001B  0149 R                   Z1       DW      L12             ; RETURN POINTERS FOR RTNS CALLED
001D  0157 R                            DW      L14             ; BEFORE STACK INITIALIZED
001F  016D R                            DW      L16
0021  0186 R                            DW      L19
0023  01BA R                            DW      L24
0025  20 4B 42                 F3B      DB      ' KB'
0028  0A47 R                   EX_0     DW      OFFSET  EB0
002A  0A47 R                            DW      OFFSET  EB0
002C  0ABB R                            DW      OFFSET  TOTLTPO
002E  0A84 R                   EX1      DW      OFFSET  M01
                               ;-----------------------------------------
                               ;        MESSAGE AREA FOR POST            :
                               ;-----------------------------------------
0030  45 52 52 4F 52           ERROR_ERR        DB      'ERROR' ;GENERAL ERROR PROMPT
0035  41                       MEM_ERR          DB      'A'     ;MEMORY ERROR
0036  42                       KEY_ERR          DB      'B'     ;KEYBOARD ERROR MSG
0037  43                       CASS_ERR         DB      'C'     ;CASSETTE ERROR MESSAGE
0038  44                       COM1_ERR         DB      'D'     ;ON-BOARD SERIAL PORT ERR. MSG
0039  45                       COM2_ERR         DB      'E'     ;SERIAL PORTION OF MODEM ERROR
003A  46                       ROM_ERR          DB      'F'     ;OPTIONAL GENERIC BIOS ROM ERROR
003B  47                       CART_ERR         DB      'G'     ;CARTRIDGE ERROR
003C  48                       DISK_ERR         DB      'H'     ;DISKETTE ERR
                               ;
003D                           F4       LABEL   WORD            ; PRINTER SOURCE TABLE
003D  0378                              DW      378H
003F  0278                              DW      278H
0041                           F4E      LABEL   WORD
0041                           IMASKS   LABEL      BYTE         ; INTERRUPT MASKS FOR 8259
                                                                ; INTERRUPT CONTROLLER
0041  EF                                DB         0EFH         ; MODEM INTR MASK
0042  F7                                DB         0F7H         ; SERIAL PRINTER INTR MASK
                               ;------------------------------------------
                               ; SETUP                                    :
                               ;        DISABLE NMI, MASKABLE INTS.       :
                               ;        SOUND CHIP, AND VIDEO.            :
                               ;        TURN DRIVE 0 MOTOR OFF            :
                               ;------------------------------------------
                                        ASSUME  CS:CODE,DS:ABS0,ES:NOTHING,SS:STACK
0043                           RESET    LABEL   FAR
0043  B0 00                    START:   MOV     AL,0
0045  E6 A0                             OUT     0A0H,AL         ; DISABLES NMI
0047  FE C8                             DEC     AL              ; SEND FF TO MFG_TESTER
0049  E6 10                             OUT     10H,AL
004B  E4 A0                             IN      AL,0A0H         ; RESET NMI F/F
004D  FA                                CLI                     ; DISABLES MASKABLE INTERRUPTS
                                                                ; DISABLE ATTENUATION IN SOUND CHIP
004E  B8 108F                           MOV     AX,108FH        ; REG ADDRESS IN AH, ATTENUATOR OFF
                                                                ; IN AL
0051  BA 00C0                           MOV     DX,00C0H        ; ADDRESS OF SOUND CHIP
0054  B9 0004                           MOV     CX,4            ; 4 ATTENUATORS TO DISABLE
0057  0A C4                    L1:      OR      AL,AH           ; COMBINE REG ADDRESS AND DATA
0059  EE                                OUT     DX,AL
005A  80 C4 20                          ADD     AH,20H          ; POINT TO NEXT REG
005D  E2 F8                             LOOP    L1
005F  B0 A0                             MOV     AL,WD_ENABLE+FDC_RESET  ; TURN DRIVE 0 MOTOR OFF,
                                                                ; ENABLE TIMER
0061  E6 F2                             OUT     0F2H,AL
0063  BA 03DA                           MOV     DX,VGA_CTL      ; VIDEO GATE ARRAY CONTROL
0066  EC                                IN      AL,DX           ; SYNC VGA TO ACCEPT REG
0067  B0 04                             MOV     AL,4            ; SET VGA RESET REG
0069  EE                                OUT     DX,AL           ; SELECT IT
006A  B0 01                             MOV     AL,1            ; SET ASYNC RESET
006C  EE                                OUT     DX,AL           ; RESET VIDEO GATE ARRAY
                               ;-----------------------------------------
                               ; TEST 1                                  :
                               ;        8088 PROCESSOR TEST              :
                               ; DESCRIPTION                             :
                               ;        VERIFY 8088 FLAGS, REGISTERS     :
                               ;        AND CONDITIONAL JUMPS            :
                               ;                                         ;
                               ; MFG. ERROR CODE 0001H                   :
                               ;-----------------------------------------
```

```
006D  B4 D5                         MOV    AH,0D5H          ; SET SF, CF, ZF, AND AF FLAGS ON
006F  9E                            SAHF
0070  73 4C                         JNC    L4               ; GO TO ERR ROUTINE IF CF NOT SET
0072  75 4A                         JNZ    L4               ; GO TO ERR ROUTINE IF ZF NOT SET
0074  7B 48                         JNP    L4               ; GO TO ERR ROUTINE IF PF NOT SET
0076  79 46                         JNS    L4               ; GO TO ERR ROUTINE IF SF NOT SET
0078  9F                            LAHF                    ; LOAD FLAG IMAGE TO AH
0079  B1 05                         MOV    CL,5             ; LOAD CNT REG WITH SHIFT CNT
007B  D2 EC                         SHR    AH,CL            ; SHIFT AF INTO CARRY BIT POS
007D  73 3F                         JNC    L4               ; GO TO ERR ROUTINE IF AF NOT SET
007F  B0 40                         MOV    AL,40H           ; SET THE OF FLAG ON
0081  D0 E0                         SHL    AL,1             ; SETUP FOR TESTING
0083  71 39                         JNO    L4               ; GO TO ERR ROUTINE IF OF NOT SET
0085  32 E4                         XOR    AH,AH            ; SET AH = 0
0087  9E                            SAHF                    ; CLEAR SF, CF, ZF, AND PF
0088  76 34                         JBE    L4               ; GO TO ERR ROUTINE IF CF ON
                                                            ; GO TO ERR ROUTINE IF ZF ON
008A  78 32                         JS     L4               ; GO TO ERR ROUTINE IF SF ON
008C  7A 30                         JP     L4               ; GO TO ERR ROUTINE IF PF ON
008E  9F                            LAHF                    ; LOAD FLAG IMAGE TO AH
008F  B1 05                         MOV    CL,5             ; LOAD CNT REG WITH SHIFT CNT
0091  D2 EC                         SHR    AH,CL            ; SHIFT 'AF' INTO CARRY BIT POS
0093  72 29                         JC     L4               ; GO TO ERR ROUTINE IF ON
0095  D0 E4                         SHL    AH,1             ; CHECK THAT 'OF' IS CLEAR
0097  70 25                         JO     L4               ; GO TO ERR ROUTINE IF ON
                             ;----- READ/WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
                             ;      WITH ALL ONE'S AND ZEROES'S.
0099  B8 FFFF                       MOV    AX,0FFFFH        ; SETUP ONE'S PATTERN IN AX
009C  F9                            STC
009D  8E D8                  L2:    MOV    DS,AX            ; WRITE PATTERN TO ALL REGS
009F  8C DB                         MOV    BX,DS
00A1  8E C3                         MOV    ES,BX
00A3  8C C1                         MOV    CX,ES
00A5  8E D1                         MOV    SS,CX
00A7  8C D2                         MOV    DX,SS
00A9  8B E2                         MOV    SP,DX
00AB  8B EC                         MOV    BP,SP
00AD  8B F5                         MOV    SI,BP
00AF  8B FE                         MOV    DI,SI
00B1  73 07                         JNC    L3
00B3  33 C7                         XOR    AX,DI            ; PATTERN MAKE IT THRU ALL REGS
00B5  75 07                         JNZ    L4               ; NO - GO TO ERR ROUTINE
00B7  F8                            CLC
00B8  EB E3                         JMP    L2
00BA  0B C7                  L3:    OR     AX,DI            ; ZERO PATTERN MAKE IT THRU?
00BC  74 0C                         JZ     L5               ; YES - GO TO NEXT TEST
00BE  BA 0010                L4:    MOV    DX,0010H         ; HANDLE ERROR
00C1  B0 00                         MOV    AL,0
00C3  EE                            OUT    DX,AL            ; ERROR 0001
00C4  42                            INC    DX
00C5  EE                            OUT    DX,AL
00C6  FE C0                         INC    AL
00C8  EE                            OUT    DX,AL
00C9  F4                            HLT                     ; HALT
00CA                         L5:
                             ;----------------------------------------
                             ; TEST 2                                 :
                             ;      8255 INITIALIZATION AND TEST      :
                             ; DESCRIPTION                            :
                             ;      FIRST INITIALIZE 8255 PROG.       :
                             ;      PERIPHERAL INTERFACE. PORTS A&B   :
                             ;      ARE LATCHED OUTPUT                :
                             ;      BUFFERS. C IS INPUT.              :
                             ; MFG. ERR. CODE =0002H                  :
                             ;----------------------------------------
00CA  B0 FE                         MOV    AL,0FEH          ; SEND FE TO MFG
00CC  E6 10                         OUT    10H,AL
00CE  B0 89                         MOV    AL,MODE_8255
00D0  E6 63                         OUT    CMD_PORT,AL      ; CONFIGURES I/O PORTS
00D2  2B C0                         SUB    AX,AX            ; TEST PATTERN SEED = 0000
00D4  8A C4                  L6:    MOV    AL,AH
00D6  E6 60                         OUT    PORT_A,AL        ; WRITE PATTERN TO PORT A
00D8  E4 60                         IN     AL,PORT_A        ; READ PATTERN FROM PORT A
00DA  E6 61                         OUT    PORT_B,AL        ; WRITE PATTERN TO PORT B
00DC  E4 61                         IN     AL,PORT_B        ; READ OUTPUT PORT
00DE  3A C4                         CMP    AL,AH            ; DATA AS EXPECTED?
00E0  75 06                         JNE    L7               ; IF NOT, SOMETHING IS WRONG
00E2  FE C4                         INC    AH               ; MAKE NEW DATA PATTERN
00E4  75 EE                         JNZ    L6               ; LOOP TILL 255 PATTERNS DONE
00E6  EB 05                         JMP    SHORT L8         ; CONTINUE IF DONE
00E8  B3 02                  L7:    MOV    BL,02H           ; SET ERROR FLAG (BH=00 NOW)
00EA  E9 09BC R                     JMP    E_MSG            ; GO ERROR ROUTINE
00ED  32 C0                  L8:    XOR    AL,AL            ;
00EF  E6 60                         OUT    KBPORT,AL        ; CLEAR KB PORT
00F1  E4 62                         IN     AL,PORT_C        ;
00F3  24 08                         AND    AL,00001000B     ; 64K CARD PRESENT?
00F5  B0 1B                         MOV    AL,1BH           ; PORT SETTING FOR 64K SYS
00F7  75 02                         JNZ    L9               ;
00F9  B0 3F                         MOV    AL,3FH           ; PORT SETTING FOR 128K SYS
00FB  BA 03DF                L9:    MOV    DX,PAGREG        ;
00FE  EE                            OUT    DX,AL            ;
00FF  B0 0D                         MOV    AL,00001101B     ;     INITIALIZE OUTPUT PORTS
0101  E6 61                         OUT    PORT_B,AL        ;
```

```
                               ;---------------------------------------------------------------
                               ; PART 3
                               ;       SET UP VIDEO GATE ARRAY AND 6845 TO GET MEMORY WORKING
                               ;
                               ;---------------------------------------------------------------
0103  B0 FD                            MOV.    AL,0FDH
0105  E6 10                            OUT     10H,AL          ;
0107  BA 03D4                          MOV     DX,03D4H        ; SET ADDRESS OF 6845
010A  BB F0A4 R                        MOV     BX,OFFSET VIDEO_PARMS ; POINT TO 6845 PARMS
010D  B9 0010 90                       MOV     CX,M0040        ; SET PARM LEN
0111  32 E4                            XOR     AH,AH           ; AH IS REG #
0113  8A C4                    L10:    MOV     AL,AH           ; GET 6845 REG #
0115  EE                               OUT     DX,AL
0116  42                               INC     DX              ; POINT TO DATA PORT
0117  FE C4                            INC     AH              ; NEXT REG VALUE
0119  2E: 8A 07                        MOV     AL,CS:[BX]      ; GET TABLE VALUE
011C  EE                               OUT     DX,AL           ; OUT TO CHIP
011D  43                               INC     BX              ; NEXT IN TABLE
011E  4A                               DEC     DX              ; BACK TO POINTER REG
011F  E2 F2                            LOOP    L10
                               ;    START VGA WITHOUT VIDEO ENABLED
0121  BA 03DA                          MOV     DX,VGA_CTL      ; SET ADDRESS OF VGA
0124  EC                               IN      AL,DX           ; BE SURE ADDR/DATA FLAG IS
                                                               ; IN THE PROPER STATE
0125  B9 0005                          MOV     CX,5            ; # OF REGISTERS
0128  32 E4                            XOR     AH,AH           ; AH IS REG COUNTER
012A  8A C4                    L11:    MOV     AL,AH           ; GET REG #
012C  EE                               OUT     DX,AL           ; SELECT IT
012D  32 C0                            XOR     AL,AL           ; SET ZERO FOR DATA
012F  EE                               OUT     DX,AL
0130  FE C4                            INC     AH              ; NEXT REG
0132  E2 F6                            LOOP    L11
                               ;------------------------------------------------
                               ; TEST 4                                        :
                               ;       PLANAR BOARD ROS CHECKSUM TEST          :
                               ; DESCRIPTION                                   :
                               ;       A CHECKSUM TEST IS DONE FOR EACH ROS    :
                               ;       MODULE ON THE PLANAR BOARD TO.          :
                               ;       MFG ERROR CODE =0003H MODULE AT ADDRESS :
                               ;                       F000:0000 ERROR         :
                               ;                       0004H MODULE AT ADDRESS :
                               ;                       F800:0000 ERROR         ;
                               ;------------------------------------------------
0134  B0 FC                            MOV     AL,0FCH
0136  E6 10                            OUT     10H,AL          ; MFG OUT=FC
                               ; CHECK MODULE AT F000:0 (LENGTH 32K)
0138  33 F6                            XOR     SI,SI           ; INDEX OFFSET WITHIN SEGMENT OF
                                                               ; FIRST BYTE
013A  8C C8                            MOV     AX,CS           ; SET UP STACK SEGMENT
013C  8E D0                            MOV     SS,AX
013E  8E D8                            MOV     DS,AX           ; LOAD DS WITH SEGMENT OF ADDRESS
                                                               ; SPACE OF BIOS/BASIC
0140  B9 8000                          MOV     CX,8000H        ; NUMBER OF BYTES TO BE TESTED, 32K
0143  BC 001B R                        MOV     SP,OFFSET Z1    ; SET UP STACK POINTER SO THAT
                                                               ; RETURN WILL COME HERE
0146  E9 FEEB R                        JMP     ROS_CHECKSUM    ; JUMP TO ROUTINE WHICK PERFORMS
                                                               ; CRC CHECK
0149  74 06                    L12:    JZ      L13             ; MODULE AT F000:0 OK, GO CHECK
                                                               ; OTHER MODULE AT F000:8000
014B  BB 0003                          MOV     BX,0003H        ; SET ERROR CODE
014E  E9 09BC R                        JMP     E_MSG           ; INDICATE ERROR
0151  B9 8000                  L13:    MOV     CX,8000H        ; LOAD COUNT (SI POINTING TO START
0154  E9 FEEB R                        JMP     ROS_CHECKSUM    ; OF NEXT MODULE AT THIS POINT)
0157  74 06                    L14:    JZ      L15             ; PROCEED IF NO ERROR
0159  BB 0004                          MOV     BX,0004H        ; INDICATE ERROR
015C  E9 09BC R                        JMP     E_MSG           ;
015F                           L15:
                               ;------------------------------------------------
                               ; TEST 5
                               ;       BASE  2K READ/WRITE STORAGE TEST
                               ; DESCRIPTION
                               ;       WRITE/READ/VERIFY DATA PATTERNS
                               ;       AA,55, AND 00 TO 1ST 2K OF STORAGE
                               ;       AND THE 2K JUST BELOW 64K (CRT BUFFER)
                               ;       VERIFY STORAGE ADDRESSABILITY.
                               ;       ON EXIT SET CRT PAGE TO 3. SET
                               ;       TEMPORARY STACK ALSO.
                               ;  MFG. ERROR CODE 04XX FOR SYSTEM BOARD MEM.
                               ;                  05XX FOR 64K ATTRIB. CD. MEM
                               ;                  06XX FOR ERRORS IN BOTH
                               ;                  (XX= ERROR BITS)
                               ;-----------------------------------------------:
015F  B0 FB                            MOV     AL,0FBH
0161  E6 10                            OUT     10H,AL          ; SET MFG FLAG=FB
0163  B9 0400                          MOV     CX,0400H        ; SET FOR 1K WORDS, 2K BYTES
0166  33 C0                            XOR     AX,AX
0168  8E C0                            MOV     ES,AX           ; LOAD ES WITH 0000 SEGMENT
016A  E9 0B59 R                        JMP     PODSTG
016D  75 19                    L16:    JNZ     L20             ; BAD STORAGE FOUND
016F  B0 FA                            MOV     AL,0FAH         ; MFG OUT=FA
0171  E6 10                            OUT     10H,AL
0173  B9 0400                          MOV     CX,400H         ; 1024 WORDS TO BE TESTED IN THE
                                                               ; REGEN BUFFER
0176  E4 60                            IN      AL,PORT_A       ; WHERE IS THE REGEN BUFFER?
0178  3C 1B                            CMP     AL,1BH          ; TOP OF 64K?
017A  B8 0F80                          MOV     AX,0F80H        ; SET POINTER TO THERE IF IT IS
017D  74 02                            JE      L18
017F  B4 1F                            MOV     AH,1FH          ; OR SET POINTER TO TOP OF 128K
0181  8E C0                    L18:    MOV     ES,AX
0183  E9 0B59 R                        JMP     PODSTG          ;
0186  74 23                    L19:    JZ      L23
```

```
0188  B7 04                     L20:    MOV     BH,04H          ; ERROR 04....
018A  E4 62                             IN      AL,PORT_C       ; GET CONFIG BITS
018C  24 08                             AND     AL,00001000B    ; TEST FOR ATTRIB CARD PRESENT
018E  74 06                             JZ      L21             ; WORRY ABOUT ODD/EVEN IF IT IS
0190  8A D9                             MOV     BL,CL
0192  0A DD                             OR      BL,CH           ; COMBINE ERROR BITS IF IT ISN'T
0194  EB 12                             JMP     SHORT L22       ;
0196  80 FC 02                  L21:    CMP     AH,02           ; EVEN BYTE ERROR? ERR 04XX
0199  8A D9                             MOV     BL,CL
019B  74 0B                             JE      L22
019D  FE C7                             INC     BH              ; MAKE INTO 05XX ERR
019F  0A DD                             OR      BL,CH           ; MOVE AND POSSIBLY COMBINE
                                                                ; ERROR BITS
01A1  80 FC 01                          CMP     AH,1            ; ODD BYTE ERROR
01A4  74 02                             JE      L22
01A6  FE C7                             INC     BH              ; MUST HAVE BEEN BOTH
                                                                ; - MAKE INTO 06XX
01A8  E9 09BC R                 L22:    JMP     E_MSG           ; JUMP TO ERROR OUTPUT ROUTINE
                                ; RETEST HIGH 2K USING B8000 ADDRESS PATH
01AB  B0 F9                     L23:    MOV     AL,0F9H         ; MFG OUT =F9
01AD  E6 10                             OUT     10H,AL
01AF  B9 0400                           MOV     CX,0400H        ; 1K WORDS
01B2  B8 B880                           MOV     AX,0B880H       ; POINT TO AREA JUST TESTED WITH
                                                                ; DIRECT ADDRESSING
01B5  8E C0                             MOV     ES,AX
01B7  E9 0B59 R                         JMP     PODSTG
01BA  74 06                     L24:    JZ      L25
01BC  BB 0005                           MOV     BX,0005H        ; ERROR 0005
01BF  E9 09BC R                         JMP     E_MSG
                                ;----- SETUP STACK SEG AND SP
01C2  B8 0030                   L25:    MOV     AX,0030H        ; GET STACK VALUE
01C5  8E D0                             MOV     SS,AX           ; SET THE STACK UP
01C7  BC 0100 R                         MOV     SP,OFFSET TOS   ; STACK IS READY TO GO
01CA  33 C0                             XOR     AX,AX           ; SET UP DATA SEG
01CC  8E D8                             MOV     DS,AX
                                ;----- SETUP CRT PAGE
01CE  C7 06 0462 R 0007                 MOV     DATA_WORD[ACTIVE_PAGE-DATA],07
                                ;----- SET PRELIMINARY MEMORY SIZE WORD
01D4  BB 0040                           MOV     BX,64
01D7  E4 62                             IN      AL,PORT_C       ;
01D9  24 08                             AND     AL,08H          ; 64K CARD PRESENT?
01DB  B0 1B                             MOV     AL,1BH          ; PORT SETTING FOR 64K SYSTEM
01DD  75 05                             JNZ     L26             ; SET TO 64K IF NOT
01DF  83 C3 40                          ADD     BX,64           ; ELSE SET FOR 128K
01E2  B0 3F                             MOV     AL,3FH          ; PORT SETTING FOR 128K SYSTEM
01E4  89 1E 0415 R              L26:    MOV     DATA_WORD[TRUE_MEM-DATA],BX
01E8  A2 048A R                         MOV     DATA_AREA[PAGDAT-DATA],AL
                                ;-----------------------------------------------------
                                ; PART 6
                                ;       INTERRUPTS
                                ; DESCRIPTION
                                ;       32 INTERRUPTS ARE INITIALIZED TO POINT TO A
                                ;       DUMMY HANDLER. THE BIOS INTERRUPTS ARE LOADED.
                                ;       DIAGNOSTIC INTERRUPTS ARE LOADED
                                ;       SYSTEM CONFIGURATION WORD IS PUT IN MEMORY.
                                ;       THE DUMMY INTERRUPT HANDLER RESIDES HERE.
                                ;-----------------------------------------------------
                                        ASSUME  DS:XXDATA
01EB  B8 ---- R                         MOV     AX,XXDATA
01EE  8E D8                             MOV     DS,AX
01F0  C6 06 0005 R F8                   MOV     MFG_TST,0F8H    ; SET UP MFG CHECKPOINT FROM THIS
                                                                ; POINT
01F5  E8 E6D8 R                         CALL    MFG_UP          ; UPDATE MFG CHECKPOINT
01F8  C7 06 0022 R 0A61 R               MOV     MFG_RTN,OFFSET MFG_OUT
01FE  8C C8                             MOV     AX,CS
0200  A3 0024 R                         MOV     MFG_RTN+2,AX    ; SET DOUBLEWORD POINTER TO MFG.
                                                                ; ERROR OUTPUT ROUTINE SO DIAGS.
                                                                ; DON'T HAVE TO DUPLICATE CODE
                                        ASSUME  CS:CODE,DS:ABS0
0203  B8 0000                           MOV     AX,0
0206  8E D8                             MOV     DS,AX
                                ;----- SET UP THE INTERRUPT VECTORS TO TEMP INTERRUPT
0208  B9 00FF                           MOV     CX,255          ; FILL ALL INTERRUPTS
020B  2B FF                             SUB     DI,DI           ; FIRST INTERRUPT LOCATION IS 0000
020D  8E C7                             MOV     ES,DI           ; SET ES=0000 ALSO
020F  B8 F815 R                 D3:     MOV     AX,OFFSET D11   ; MOVE ADDR OF INTR PROC TO TBL
0212  AB                                STOSW
0213  8C C8                             MOV     AX,CS           ; GET ADDR OF INTR PROC SEG
0215  AB                                STOSW
0216  E2 F7                             LOOP    D3              ; VECTBLO
0218  C7 06 0124 R 109D R               MOV     EXST,OFFSET EXTAB ; SET UP EXT. SCAN TABLE
                                ; SET UP BIOS INTERRUPTS
021E  BF 0040 R                         MOV     DI,OFFSET VIDEO_INT ; SET UP VIDEO INT
0221  0E                                PUSH    CS
0222  1F                                POP     DS              ; PLACE CS IN DS
0223  BE FF03 R                         MOV     SI,OFFSET VECTOR_TABLE+16
0226  B9 0010                           MOV     CX,16
0229  A5                        D4:     MOVSW                   ; MOVE INTERRUPT VECTOR TO LOW
                                                                ; MEMORY
022A  47                                INC     DI
022B  47                                INC     DI              ; POINT TO NEXT VECTOR ENTRY
022C  E2 FB                             LOOP    D4              ; REPEAT FOR ALL 16 BIOS INTERRUPTS
                                ; SET UP DIAGNOSTIC INTERRUPTS
022E  BF 0200                           MOV     DI,0200H        ; START WITH INT. 80H
0231  BE 4000                           MOV     SI,DIAG_TABLE_PTR ; POINT TO ENTRY POINT TABLE
0234  B9 0010                           MOV     CX,16           ; 16 ENTRIES
0237  A5                        D5:     MOVSW                   ; MOVE INTERRUPT VECTOR TO LOW
                                                                ; MEMORY
```

```
0238  47                                 INC     DI
0239  47                                 INC     DI                ; POINT TO NEXT VECTOR ENTRY
023A  E2 FB                              LOOP    D5                ; REPEAT FOR ALL 16 BIOS INTERRUPTS
023C  8E D9                              MOV     DS,CX             ; SET DS TO ZERO
023E  C7 06 0204 R 1B63 R                MOV     INT81,OFFSET LOCATEI
0244  C7 06 0208 R 1A2A R                MOV     INT82,OFFSET PRNT3
024A  C7 06 0224 R 1BA5 R                MOV     INT89,OFFSET JOYSTICK

                                ;----- SET UP DEFAULT EQUIPMENT DETERMINATION WORD
                                ;        BIT 15,14 = NUMBER OF PRINTERS ATTACHED
                                ;        BIT 13 = 1 = SERIAL PRINTER PRESENT
                                ;        BIT 12 = GAME I/O ATTACHED
                                ;        BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
                                ;        BIT 8 = DMA (0=DMA PRESENT, 1=NO DMA ON SYSTEM
                                ;        BIT 7,6 = NUMBER OF DISKETTE DRIVES
                                ;                00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1
                                ;        BIT 5,4 = INITIAL VIDEO MODE
                                ;                        00 - UNUSED
                                ;                        01 - 40X25 BW USING COLOR CARD
                                ;                        10 - 80X25 BW USING COLOR CARD
                                ;                        11 - 80X25 BW USING BW CARD
                                ;        BIT 3,2 = PLANAR RAM SIZE (10=48K,11=64K)
                                ;        BIT 1 NOT USED
                                ;        BIT 0 = 1 (IPL DISKETTE INSTALLED)
                                ;---------------------------------------------------
                                         ASSUME  CS:CODE,DS:ABS0
0250  BB 1118                            MOV     BX,1118H          ;DEFAULT GAMEIO,40X25,NO DMA,48K ON
                                                                   ; PLANAR
0253  E4 62                              IN      AL,PORT_C
0255  24 08                              AND     AL,08H            ; 64K CARD PRESENT
0257  75 03                              JNZ     D55               ; NO, JUMP
0259  80 CB 04                           OR      BL,4              ; SET 64K ON PLANAR
025C  89 1E 0410 R              D55:     MOV     DATA_WORD[EQUIP_FLAG-DATA],BX
                                ;----------------------------------------------------------------
                                ; TEST 7
                                ;        INITIALIZE AND TEST THE 8259 INTERRUPT CONTROLLER CHIP
                                ; MFG ERR. CODE 07XX (XX=00, DATA PATH OR INERNAL FAILURE,
                                ;        XX=ANY OTHER BITS ON=UNEPECTED INTERRUPTS
                                ;----------------------------------------------------------------
0260  E8 E6D8 R                          CALL    MFG_UP            ; MFG CODE=F7
                                         ASSUME  DS:ABS0,CS:CODE
0263  B0 13                              MOV     AL,13H            ; ICW1 - RESET EDGE SENSE CIRCUIT,
                                                                   ;SET SINGLE 8259 CHIP AND ICW4 READ
0265  E6 20                              OUT     INTA00,AL
0267  B0 08                              MOV     AL,8              ; ICW2 - SET INTERRUPT TYPE 8 (8-F)
0269  E6 21                              OUT     INTA01,AL
026B  B0 09                              MOV     AL,9              ; ICW4 - SET BUFFERED MODE/SLAVE
                                                                   ;   AND 8086 MODE
026D  E6 21                              OUT     INTA01,AL
                                ;----------------------------------------------------------------
                                ;        TEST ABILITY TO WRITE/READ THE MASK REGISTER
                                ;----------------------------------------------------------------
026F  B0 00                              MOV     AL,0              ; WRITE ZEROES TO IMR
0271  8A D8                              MOV     BL,AL             ; PRESET ERROR INDICATOR
0273  E6 21                              OUT     INTA01,AL         ; DEVICE INTERRUPTS ENABLED
0275  E4 21                              IN      AL,INTA01         ; READ IMR
0277  0A C0                              OR      AL,AL             ; IMR = 0?
0279  75 18                              JNZ     GERROR            ; NO - GO TO ERROR ROUTINE
027B  B0 FF                              MOV     AL,0FFH           ; DISABLE DEVICE INTERRUPTS
027D  E6 21                              OUT     INTA01,AL         ; WRITE ONES TO IMR
027F  E4 21                              IN      AL,INTA01         ; READ IMR
0281  04 01                              ADD     AL,1              ; ALL IMR BITS ON?
                                                                   ; (ADD SHOULD PRODUCE 0)
0283  75 0E                              JNZ     GERROR            ; NO - GO TO ERROR ROUTINE
                                ;----------------------------------------------------------------
                                ;        CHECK FOR HOT INTERRUPTS
                                ;----------------------------------------------------------------
                                ;        INTERRUPTS ARE MASKED OFF.  NO INTERRUPTS SHOULD OCCUR.
0285  FB                                 STI                       ; ENABLE EXTERNAL INTERRUPTS
0286  B9 0050                            MOV     CX,50H
0289  E2 FE                     HOT1:    LOOP    HOT1              ; WAIT FOR ANY INTERRUPTS
028B  8A 1E 0484 R                       MOV     BL,DATA_AREA[INTR_FLAG-DATA]  ; DID ANY INTERRUPTS
                                                                   ; OCCUR?
028F  0A DB                              OR      BL,BL
0291  74 05                              JZ      END_TESTG         ; NO - GO TO NEXT TEST
0293  B7 07                     GERROR:  MOV     BH,07H            ; SET 07 SECTION OF ERROR MSG
0295  E9 09BC R                          JMP     E_MSG
0298                            END_TESTG:
                                ; FIRE THE DISKETTE WATCHDOG TIMER
0298  B0 E0                              MOV     AL,WD_ENABLE+WD_STROBE+FDC_RESET
029A  E6 F2                              OUT     0F2H,AL
029C  B0 A0                              MOV     AL,WD_ENABLE+FDC_RESET
029E  E6 F2                              OUT     0F2H,AL
                                         ASSUME  CS:CODE,DS:ABS0
                                ;------------------------------------------------------------------
                                ;        8253 TIMER CHECKOUT
                                ; DESCRIPTION
                                ;        VERIFY THAT THE TIMERS (0, 1, AND 2) FUNCTION PROPERLY.
                                ;        THIS INCLUDES CHECKING FOR STUCK BITS IN ALL THE TIMERS,
                                ;        THAT TIMER 1 RESPONDS TO TIMER 0 OUTPUTS, THAT TIMER 0
                                ;        INTERRUPTS WHEN IT SHOULD, AND THAT TIMER 2'S OUTPUT WORKS
                                ;        AS IT SHOULD.
                                ;        THERE ARE 7 POSSIBLE ERRORS DURING THIS CHECKOUT.
                                ;        BL VALUES FOR THE CALL TO E_MSG INCLUDE:
                                ;        0)  STUCK BITS IN TIMER 0
                                ;        1)  TIMER 1 DOES NOT RESPOND TO TIMER 0 OUTPUT
                                ;        2)  TIMER 0 INTERRUPT DOES NOT OCCUR
                                ;        3)  STUCK BITS IN TIMER 1
                                ;        4)  TIMER 2 OUTPUT INITIAL VALUE IS NOT LOW
                                ;        5)  STUCK BITS IN TIMER 2
                                ;        6)  TIMER 2 OUTPUT DOES NOT GO HIGH ON TERMINAL COUNT
                                ;-----------------------------------------------------------------
                                ;       INITIALIZE TIMER 1 AND TIMER 0 FOR TEST
                                ;-----------------------------------------------------------------
02A0  E8 E6D8 R                         CALL    MFG_UP                  ; MFG CKPOINT=F6
02A3  B8 0176                           MOV     AX,0176H                ; SET TIMER 1 TO MODE 3 BINARY
02A6  BB FFFF                           MOV     BX,0FFFFH               ; INITIAL COUNT OF FFFF
02A9  E8 FFE0 R                         CALL    INIT_TIMER              ; INITIALIZE TIMER 1
02AC  B8 0036                           MOV     AX,0036H                ; SET TIMER 0 TO MODE 3 BINARY
                                                                        ; INITIAL COUNT OF FFFF
02AF  E8 FFE0 R                         CALL    INIT_TIMER              ; INITIALIZE TIMER 0
                                ;-----------------------------------------------------------------
                                ;       SET BIT 5 OF PORT A0 SO TIMER 1 CLOCK WILL BE PULSED BY THE
                                ;       TIMER 0 OUTPUT RATHER THAN THE SYSTEM CLOCK.
                                ;-----------------------------------------------------------------
02B2  B0 20                             MOV     AL,00100000B
02B4  E6 A0                             OUT     0A0H,AL
                                ;-----------------------------------------------------------------
                                ;       CHECK IF ALL BITS GO ON AND OFF IN TIMER 0 (CHECK FOR STUCK
                                ;          BITS)
                                ;-----------------------------------------------------------------
02B6  B4 00                             MOV     AH,0                    ; TIMER 0
02B8  E8 036C R                         CALL    BITS_ON_OFF             ; LET SUBROUTINE CHECK IT
02BB  73 05                             JNB     TIMER1_NZ               ;NO STUCK BITS (CARRY FLAG NOT SET)
02BD  B3 00                             MOV     BL,0                    ; STUCK BITS IN TIMER 0
02BF  E9 0362 R                         JMP     TIMER_ERROR
                                ;-----------------------------------------------------------------
                                ;       SINCE TIMER 0 HAS COMPLETED AT LEAST ONE COMPLETE CYCLE,
                                ;       TIMER 1 SHOULD BE NON-ZERO.  CHECK THAT THIS IS THE CASE.
                                ;-----------------------------------------------------------------
02C2                            TIMER1_NZ:
02C2  E4 41                             IN      AL,TIMER+1              ; READ LSB OF TIMER 1
02C4  8A E0                             MOV     AH,AL                   ; SAVE LSB
02C6  E4 41                             IN      AL,TIMER+1              ; READ MSB OF TIMER 1
02C8  3D FFFF                           CMP     AX,0FFFFH               ; STILL FFFF?
02CB  75 05                             JNE     TIMER0_INTR             ; NO - TIMER 1 HAS BEEN BUMPED
02CD  B3 01                             MOV    BL,1                     ; TIMER 1 WAS NOT BUMPED BY TIMER 0
02CF  E9 0362 R                         JMP     TIMER_ERROR
                                ;-----------------------------------------------------------------
                                ;       CHECK FOR TIMER 0 INTERRUPT
                                ;-----------------------------------------------------------------
02D2                            TIMER0_INTR:
02D2  FB                                STI                             ; ENABLE MASKABLE EXT INTERRUPTS
02D3  E4 21                             IN      AL,INTA01
02D5  24 FE                             AND     AL,0FEH                 ; MASK ALL INTRS EXCEPT LVL 0
02D7  20 06 0484 R                      AND     DATA_AREA[INTR_FLAG-DATA],AL ; CLEAR INT RECEIVED
02DB  E6 21                             OUT     INTA01,AL               ; WRITE THE 8259 IMR
02DD  B9 FFFF                           MOV     CX,0FFFFH               ; SET LOOP COUNT
02E0                            WAIT_INTR_LOOP:
02E0  F6 06 0484 R 01                   TEST    DATA_AREA[INTR_FLAG-DATA],1  ; TIMER 0 INT OCCUR?
02E5  75 06                             JNE     RESET_INTRS             ; YES - CONTINUE
02E7  E2 F7                             LOOP    WAIT_INTR_LOOP          ; WAIT FOR INTR FOR SPECIFIED TIME
02E9  B3 02                             MOV     BL,2                    ; TIMER 0 INTR DIDN'T OCCUR
02EB  EB 75                             JMP     SHORT TIMER_ERROR
                                ;-----------------------------------------------------------------
                                ;       HOUSEKEEPING FOR TIMER 0 INTERRUPTS
                                ;-----------------------------------------------------------------
02ED                            RESET_INTRS:
02ED  FA                                CLI
                                ; SET TIMER INT. TO POINT TO MFG. HEARTBEAT ROUTINE IF IN MFG MODE
02EE  BA 0201                           MOV     DX,201H
02F1  EC                                IN      AL,DX                   ; GET MFG. BITS
02F2  24 F0                             AND     AL,0F0H
02F4  3C 10                             CMP     AL,10H                  ; SYS TEST MODE?
02F6  74 04                             JE      D6
02F8  0A C0                             OR      AL,AL                   ; OR BURN-IN MODE
02FA  75 11                             JNZ     TIME_1
02FC  C7 06 0020 R 188D R       D6:     MOV     INT_PTR,OFFSET MFG_TICK ; SET TO POINT TO MFG.
                                                                        ; ROUTINE
0302  C7 06 0070 R 188D R               MOV     INT1C_PTR,OFFSET MFG_TICK ; ALSO SET USER TIMER INT
                                                                        ; FOR DIAGS. USE
0308  B0 FE                             MOV     AL,0FEH
030A  E6 21                             OUT     INTA01,AL
030C  FB                                STI
                                ;-----------------------------------------------------------------
                                ;       RESET D5  OF PORT A0 SO THAT THE TIMER 1 CLOCK WILL BE
                                ;       PULSED BY THE SYSTEM CLOCK.
                                ;-----------------------------------------------------------------
030D  B0 00                     TIME_1: MOV     AL,0                    ; MAKE AL = 00
030F  E6 A0                             OUT     0A0H,AL
                                ;-----------------------------------------------------------------
                                ;       CHECK FOR STUCK BITS IN TIMER 1
                                ;-----------------------------------------------------------------
0311  B4 01                             MOV     AH,1                    ; TIMER 1
0313  E8 036C R                         CALL    BITS_ON_OFF
0316  73 04                             JNB     TIMER2_INIT             ; NO STUCK BITS
0318  B3 03                             MOV     BL,3                    ; STUCK BITS IN TIMER 1
031A  EB 46                             JMP     SHORT TIMER_ERROR
                                ;-----------------------------------------------------------------
                                ;       INITIALIZE TIMER 2
                                ;-----------------------------------------------------------------
031C                            TIMER2_INIT:
031C  B8 02B6                           MOV     AX,02B6H                ; SET TIMER 2 TO MODE 3 BINARY
031F  BB FFFF                           MOV     BX,0FFFFH               ; INITIAL COUNT
0322  E8 FFE0 R                         CALL    INIT_TIMER
                                ;-----------------------------------------------------------------
                                ;       SET PB0 OF PORT_B OF 8255 (TIMER 2 GATE)
                                ;-----------------------------------------------------------------
0325  E4 61                             IN      AL,PORT_B               ; CURRENT STATUS
0327  0C 01                             OR      AL,00000001B            ; SET BIT 0 - LEAVE OTHERS ALONE
0329  E6 61                             OUT     PORT_B,AL
```

```
                                 ;----------------------------------------------------------------
                                 ;       CHECK FOR STUCK BITS IN TIMER 2
                                 ;----------------------------------------------------------------
032B  B4 02                              MOV     AH,2              ; TIMER 2
032D  E8 036C R                          CALL    BITS_ON_OFF
0330  73 04                              JNB     REINIT_T2         ; NO STUCK BITS
0332  B3 05                              MOV     BL,5              ; STUCK BITS IN TIMER 2
0334  EB 2C                              JMP     SHORT TIMER_ERROR
                                 ;----------------------------------------------------------------
                                 ;       RE_INITIALIZE TIMER 2 WITH MODE 0 AND A SHORT COUNT
                                 ;----------------------------------------------------------------
0336                             REINIT_T2:
                                 ; DROP GATE TO TIMER 2
0336  E4 61                              IN      AL,PORT_B         ; CURRENT STATUS
0338  24 FE                              AND     AL,11111110B      ; RESET BIT 0 - LEAVE OTHERS ALONE
033A  E6 61                              OUT     PORT_B,AL
033C  B8 02B0                            MOV     AX,02B0H          ; SET TIMER 2 TO MODE 0 BINARY
033F  BB 000A                            MOV     BX,000AH          ; INITIAL COUNT OF 10
0342  E8 FFE0 R                          CALL    INIT_TIMER
                                 ;----------------------------------------------------------------
                                 ;       CHECK PC5 OF PORT_C OF 8255 TO SEE IF THE OUTPUT OF TIMER 2
                                 ;       IS LOW
                                 ;----------------------------------------------------------------
0345  E4 62                              IN      AL,PORT_C         ; CURRENT STATUS
0347  24 20                              AND     AL,00100000B      ; MASK OFF OTHER BITS
0349  74 04                              JZ      CK2_ON            ; IT'S LOW
034B  B3 04                              MOV     BL,4              ; PC5 OF PORT_C WAS HIGH WHEN IT
034D  EB 13                              JMP     SHORT TIMER_ERROR ; SHOULD HAVE BEEN LOW
                                 ; TURN GATE BACK ON
034F  E4 61                      CK2_ON: IN      AL,PORT_B         ; CURRENT STATUS
0351  0C 01                              OR      AL,00000001B      ; SET BIT 0 - LEAVE OTHERS ALONE
0353  E6 61                              OUT     PORT_B,AL
                                 ;----------------------------------------------------------------
                                 ;       CHECK PC5 OF PORT_C TO SEE IF THE OUTPUT OF TIMER 2 GOES
                                 ;       HIGH
                                 ;----------------------------------------------------------------
0355  B9 000A                            MOV     CX,000AH          ; WAIT FOR OUTPUT GO HIGH, SHOULD
0358  E2 FE                      CK2_LO: LOOP    CK2_LO            ; BE LONGER THAN INITIAL COUNT
035A  E4 62                              IN      AL,PORT_C         ; CURRENT STATUS
035C  24 20                              AND     AL,00100000B      ; MASK OFF ALL OTHER BITS
035E  75 57                              JNZ     POD13_END         ; IT'S HIGH - WE'RE DONE!
0360  B3 06                              MOV     BL,6              ; TIMER 2 OUTPUT DID NOT GO HIGH
                                 ;----------------------------------------------------------------
                                 ;       8253 TIMER ERROR OCCURRED.  SET BH WITH MAJOR ERROR
                                 ;       INDICATOR AND CALL E_MSG TO INFORM THE SYSTEM OF THE ERROR.
                                 ;       (BL ALREADY CONTAINS THE MINOR ERROR INDICATOR TO TELL
                                 ;       WHICH PART OF THE TEST FAILED.)
                                 ;----------------------------------------------------------------
0362                             TIMER_ERROR:
0362  B7 08                              MOV     BH,8              ; TIMER ERROR INDICATOR
0364  E8 09BC R                          CALL    E_MSG
0367  EB 4E                              JMP     SHORT POD13_END
                                 ;----------------------------------------------------------------
                                 ;       BITS ON/OFF SUBROUTINE - USED FOR DETERMINING IF A
                                 ;       PARTICULAR TIMER'S BITS GO ON AND OFF AS THEY SHOULD.
                                 ;       THIS ROUTINE ASSUMES THAT THE TIMER IS USING BOTH THE LSB
                                 ;       AND THE MSB.
                                 ;   CALLING PARAMETER:
                                 ;       (AH) = TIMER NUMBER (0, 1, OR 2)
                                 ;   RETURNS:
                                 ;       (CF) = 1 IF FAILED
                                 ;       (CF) = 0 IF PASSED
                                 ;       REGISTERS AX, BX, CX, DX, DI, AND SI ARE ALTERED.
                                 ;----------------------------------------------------------------
0369                             LATCHES LABEL   BYTE
0369  00                                 DB      00H               ; LATCH MASK FOR TIMER 0
036A  40                                 DB      40H               ; LATCH MASK FOR TIMER 1
036B  80                                 DB      80H               ; LATCH MASK FOR TIMER 2
                                 ;
036C                             BITS_ON_OFF     PROC  NEAR
036C  33 DB                              XOR     BX,BX             ; INITIALIZE BX REGISTER
036E  33 F6                              XOR     SI,SI             ; 1ST PASS - SI = 0
0370  BA 0040                            MOV     DX,TIMER          ; BASE PORT ADDRESS FOR TIMERS
0373  02 D4                              ADD     DL,AH
0375  BF 0369 R                          MOV     DI,OFFSET LATCHES ; SELECT LATCH MASK
0378  32 C0                              XOR     AL,AL             ; CLEAR AL
037A  86 C4                              XCHG    AL,AH             ; AH -> AL
037C  03 F8                              ADD     DI,AX             ; TIMER LATCH MASK INDEX
                                 ; 1ST PASS - CHECKS FOR ALL BITS TO COME ON
                                 ; 2ND PASS - CHECKS FOR ALL BITS TO GO OFF
037E                             OUTER_LOOP:
037E  B9 0008                            MOV     CX,8              ; OUTER LOOP COUNTER
0381                             INNER_LOOP:
0381  51                                 PUSH    CX                ; SAVE OUTER LOOP COUNTER
0382  B9 FFFF                            MOV     CX,0FFFFH         ; INNER LOOP COUNTER
0385                             TST_BITS:
0385  2E: 8A 05                          MOV     AL,CS:[DI]        ; TIMER LATCH MASK
0388  E6 43                              OUT     TIM_CTL,AL        ; LATCH TIMER
038A  50                                 PUSH    AX                ; PAUSE
038B  58                                 POP     AX
038C  EC                                 IN      AL,DX             ; READ TIMER LSB
038D  0B F6                              OR      SI,SI
038F  75 0D                              JNE     SECOND            ; SECOND PASS
0391  0C 01                              OR      AL,01H            ; TURN LS BIT ON
0393  0A D8                              OR      BL,AL             ; TURN 'ON' BITS ON
0395  EC                                 IN      AL,DX             ; READ TIMER MSB
0396  0A F8                              OR      BH,AL             ; TURN 'ON' BITS ON
0398  81 FB FFFF                         CMP     BX,0FFFFH         ; ARE ALL TIMER BITS ON?
039C  EB 07                              JMP     SHORT TST_CMP     ; DON'T CHANGE FLAGS
```

```
039E                      SECOND:
039E  22 D8                       AND     BL,AL           ; CHECK FOR ALL BITS OFF
03A0  EC                          IN      AL,DX           ; READ MSB
03A1  22 F8                       AND     BH,AL           ; TURN OFF BITS
03A3  0B DB                       OR      BX,BX           ; ALL OFF?
03A5                      TST_CMP:
03A5  74 07                       JE      CHK_END         ; YES - SEE IF DONE
03A7  E2 DC                       LOOP    TST_BITS        ; KEEP TRYING
03A9  59                          POP     CX              ; RESTORE OUTER LOOP COUNTER
03AA  E2 D5                       LOOP    INNER_LOOP      ; TRY AGAIN
03AC  F9                          STC                     ; ALL TRIES EXHAUSTED - FAILED TEST
03AD  C3                          RET
03AE                      CHK_END:
03AE  59                          POP     CX              ; POP FORMER OUTER LOOP COUNTER
03AF  46                          INC     SI
03B0  83 FE 02                    CMP     SI,2
03B3  75 C9                       JNE     OUTER_LOOP      ; CHECK FOR ALL BITS TO GO OFF
03B5  F8                          CLC                     ; TIMER BITS ARE WORKING PROPERLY
03B6  C3                          RET
03B7                      BITS_ON_OFF     ENDP
03B7                      POD13_END:
                          ;-----------------------------------------------------
                          ;               CRT ATTACHMENT TEST
                          ;
                          ; 1. INIT CRT TO 40X25 - BW
                          ; 2. CHECK FOR VERTICAL AND VIDEO ENABLES, AND CHECK
                          ;    TIMING OF SAME
                          ; 3. CHECK VERTICAL INTERRUPT
                          ; 4. CHECK RED, BLUE, GREEN, AND INTENSIFY DOTS
                          ; 5. INIT TO 40X25 - COLOR
                          ;   MFG. ERROR CODE 09XX (XX-SEE COMMENETS IN CODE)
                          ;-----------------------------------------------------
= A0AC                    MAVT    EQU     0A0ACH          ; MAXIMUM TIME FOR VERT/VERT
                                                          ; (NOMINAL + 10%)
= C460                    MIVT    EQU     0C460H          ; MINIMUM TIME FOR VERT/VERT
                                                          ; (NOMINAL - 10%)
                          ;  NOMINAL TIME IS 8286H FOR 60 hz.
= 00C8                    EPF     EQU     200             ; NUMBER OF ENABLES PER FRAME

03B7  E8 E6D8 R                   CALL    MFG_UP          ; MFG CHECKPOINT= F5
03BA  FA                          CLI
03BB  B0 70                       MOV     AL,01110000B    ; SET TIMER 1 TO MODE 0
03BD  E6 43                       OUT     TIM_CTL,AL
03BF  B9 8000                     MOV     CX,8000H
03C2  E2 FE               Q1:     LOOP    Q1              ; WAIT FOR MODE SET TO "TAKE"
03C4  B0 00                       MOV     AL,00H
03C6  E6 41                       OUT     TIMER+1,AL      ; SEND FIRST BYTE TO TIMER
03C8  2B C0                       SUB     AX,AX           ; SET MODE 40X25 - BW
03CA  CD 10                       INT     10H
03CC  B8 0507                     MOV     AX,0507H        ; SET TO VIDEO PAGE 7
03CF  CD 10                       INT     10H             ;
03D1  BA 03DA                     MOV     DX,03DAH        ; SET ADDRESSING TO VIDEO ARRAY
03D4  2B C9                       SUB     CX,CX           ;
                          ; LOOK FOR VERTICAL
03D6  EC                  Q2:     IN      AL,DX           ; GET STATUS
03D7  A8 08                       TEST    AL,00001000B    ; VERTICAL THERE YET?
03D9  75 06                       JNE      Q3             ; CONTINUE IF IT IS
03DB  E2 F9                       LOOP    Q2              ; KEEP LOOKING TILL COUNT EXHAUSTED
03DD  B3 00                       MOV     BL,00           ;
03DF  EB 4C                       JMP     SHORT Q115      ; NO VERTICAL = ERROR 0900
                          ; GOT VERTICAL - START TIMER
03E1  32 C0               Q3:     XOR     AL,AL           ;
03E3  E6 41                       OUT     TIMER+1,AL      ; SEND 2ND BYTE TO TIMER TO START
03E5  2B DB                       SUB     BX,BX           ; INIT. ENABLE COUNTER
                          ; WAIT FOR VERTICAL TO GO AWAY
03E7  33 C9                       XOR     CX,CX
03E9  EC                  Q4:     IN      AL,DX           ; GET STATUS
03EA  A8 08                       TEST    AL,00001000B    ; VERTICAL STILL THERE?
03EC  74 06                       JZ      Q5              ; CONTINUE IF IT'S GONE
03EE  E2 F9                       LOOP    Q4              ; KEEP LOOKING TILL COUNT EXHAUSTED
03F0  B3 01                       MOV     BL,01H          ;
03F2  EB 39                       JMP     SHORT Q115      ; VERTICAL STUCK ON = ERROR 0901
                          ; NOW START LOOKING FOR ENABLE TRANSITIONS
03F4  2B C9               Q5:     SUB     CX,CX
03F6  EC                  Q6:     IN      AL,DX           ; GET STATUS
03F7  A8 01                       TEST    AL,00000001B    ; ENABLE ON YET?
03F9  75 0A                       JNE     Q7              ; GO ON IF IT IS
03FB  A8 08                       TEST    AL,00001000B    ; VERTICAL ON AGAIN?
03FD  75 22                       JNE     Q11             ; CONTINUE IF IT IS
03FF  E2 F5                       LOOP    Q6              ; KEEP LOOKING IF NOT
0401  B3 02                       MOV     BL,02H          ;
0403  EB 28                       JMP     SHORT Q115      ; ENABLE STUCK OFF = ERROR 0902
                          ; MAKE SURE VERTICAL WENT OFF WITH ENABLE GOING ON
0405  A8 08               Q7:     TEST    AL,00001000B    ; VERTICAL OFF?
0407  74 04                       JZ      Q8              ; GO ON IF IT IS
0409  B3 03                       MOV     BL,03H          ;
040B  EB 20                       JMP     SHORT Q115      ; VERTICAL STUCK ON = ERROR 0903
                          ; NOW WAIT FOR ENABLE TO GO OFF
040D  2B C9               Q8:     SUB     CX,CX
040F  EC                  Q9:     IN      AL,DX           ; GET STATUS
0410  A8 01                       TEST    AL,00000001B    ; ENABLE OFF YET?
0412  74 06                       JE      Q10             ; PROCEED IF IT IS
0414  E2 F9                       LOOP    Q9              ; KEEP LOOKING IF NOT YET LOW
0416  B3 04                       MOV     BL,04H          ;
0418  EB 13                       JMP     SHORT Q115      ; ENABLE STUCK ON = ERROR 0904
                          ; ENABLE HAS TOGGLED, BUMP COUNTER AND TEST FOR NEXT VERTICAL
041A  43                  Q10:    INC     BX              ; BUMP ENABLE COUNTER
041B  74 04                       JZ      Q11             ; IF COUNTER WRAPS, ERROR
041D  A8 08                       TEST    AL,00001000B    ; DID ENABLE GO LOW BECAUSE OF
                                                          ; VERTICAL?
041F  74 D3                       JZ      Q5              ; IF NOT, LOOK FOR ANOTHER ENABLE
                                                          ;   TOGGLE
```

```
                               ; HAVE HAD COMPLETE VERTICAL-VERTICAL CYCLE, NOW TEST RESULTS
0421  B0 40                    Q11:    MOV     AL,40H          ; LATCH TIMER1
0423  E6 43                            OUT     TIM_CTL,AL      ;
0425  81 FB 00C8                       CMP     BX,EPF          ; NUMBER OF ENABLES BETWEEN
                                                               ; VERTICALS O.K.?
0429  74 04                            JE      Q12             ;
042B  B3 05                            MOV     BL,05H          ;
042D  EB 74                    Q115:   JMP     SHORT Q22       ; WRONG # ENABLES = ERROR 0905
042F  E4 41                    Q12:    IN      AL,TIMER+1      ; GET TIMER VALUE LOW
0431  8A E0                            MOV     AH,AL           ; SAVE IT
0433  90                               NOP                     ;
0434  E4 41                            IN      AL,TIMER+1      ; GET TIMER HIGH
0436  86 E0                            XCHG    AH,AL           ;
0438  FB                               STI                     ; INTERRUPTS BACK ON
0439  90                               NOP
043A  3D A0AC                          CMP     AX,MAVT         ;
043D  7D 04                            JGE     Q13             ;
043F  B3 06                            MOV     BL,06H          ;
0441  EB 60                            JMP     SHORT Q22       ; VERTICALS TOO FAR APART
                                                               ; = ERROR 0906
0443  3D C460                  Q13:    CMP     AX,MIVT         ;
0446  7E 04                            JLE     Q14             ;
0448  B3 07                            MOV     BL,07H          ;
044A  EB 57                            JMP     SHORT Q22       ; VERTICALS TOO CLOSE TOGETHER
                                                               ;   = ERROR 0907
                               ; TIMINGS SEEM O.K., NOW CHECK VERTICAL INTERRUPT (LEVEL 5)
044C  2B C9                    Q14:    SUB     CX,CX           ; SET TIMEOUT REG
044E  E4 21                            IN      AL,INTA01
0450  24 DF                            AND     AL,11011111B    ; UNMASK INT. LEVEL 5
0452  E6 21                            OUT     INTA01,AL       ;
0454  20 06 0484 R                     AND     DATA_AREA[INTR_FLAG-DATA],AL
0458  FB                               STI                     ; ENABLE INTS.
0459  F6 06 0484 R 20          Q15:    TEST    DATA_AREA[INTR_FLAG-DATA],00100000B ; SEE IF INTR.
                                                               ;    5 HAPPENED YET
045E  75 06                            JNZ     Q16             ; GO ON IF IT DID
0460  E2 F7                            LOOP    Q15             ; KEEP LOOKING IF IT DIDN'T
0462  B3 08                            MOV     BL,08H          ;
0464  EB 3D                            JMP     SHORT Q22       ; NO VERTICAL INTERRUPT
                                                               ; = ERROR 0908
0466  E4 21                    Q16:    IN      AL,INTA01       ; DISABLE INTERRUPTS FOR LEVEL 5
0468  0C 20                            OR      AL,00100000B    ;
046A  E6 21                            OUT     INTA01,AL       ;
                               ; SEE IF RED, GREEN, BLUE AND INTENSIFY DOTS WORK
                               ; FIRST, SET A LINE OF REVERSE VIDEO, INTENSIFIED BLANKS INTO VIDEO
                               ; BUFFER
046C  B8 09DB                          MOV     AX,09DBH        ; WRITE CHARS, BLOCKS
046F  BB 077F                          MOV     BX,077FH        ; PAGE 7, REVERSE VIDEO,
                                                               ;    HIGH INTENSITY
0472  B9 0028                          MOV     CX,40           ; 40 CHARACTERS
0475  CD 10                            INT     10H             ;
0477  33 C0                            XOR     AX,AX           ; START WITH BLUE DOTS
0479  2B C9                    Q17:    SUB     CX,CX           ;
047B  EE                               OUT     DX,AL           ; SET VIDEO ARRAY ADDRESS FOR DOTS
                               ; SEE IF DOT COMES ON
047C  EC                       Q18:    IN      AL,DX           ; GET STATUS
047D  A8 10                            TEST    AL,00010000B    ; DOT THERE?
047F  75 08                            JNZ     Q19             ; GO LOOK FOR DOT TO TURN OFF
0481  E2 F9                            LOOP    Q18             ; CONTINUE TESTING FOR DOT ON
0483  B3 10                            MOV     BL,10H          ;
0485  0A DC                            OR      BL,AH           ; OR IN DOT BEING TESTED
0487  EB 1A                            JMP     SHORT Q22       ; DOT NOT COMING ON = ERROR 091X
                                                               ; ( X=0, BLUE; X=1, GREEN;
                                                               ; X=2, RED; X=3, INTENSITY)
                               ; SEE IF DOT GOES OFF
0489  2B C9                    Q19:    SUB     CX,CX
048B  EC                       Q20:    IN      AL,DX           ; GET STATUS
048C  A8 10                            TEST    AL,00010000B    ; IS DOT STILL ON?
048E  74 08                            JE      Q21             ; GO ON IF DOT OFF
0490  E2 F9                            LOOP    Q20             ; ELSE, KEEP WAITING FOR DOT
                                                               ; TO GO OFF
0492  B3 20                            MOV     BL,20H          ;
0494  0A DC                            OR      BL,AH           ; OR IN DOT BEING TESTED
0496  EB 0B                            JMP     SHORT Q22       ; DOT STUCK ON = ERROR 092X
                                                               ; (X=0, BLUE; X=1, GREEN;
                                                               ;  X=2, RED; X=3, INTENSITY)
                               ; ADJUST TO POINT TO NEXT DOT
0498  FE C4                    Q21:    INC     AH              ;
049A  80 FC 04                         CMP     AH,4            ; ALL 4 DOTS DONE?
049D  74 09                            JE      Q23             ; GO END
049F  8A C4                            MOV     AL,AH
04A1  EB D6                            JMP     Q17             ; GO LOOK FOR ANOTHER DOT
04A3  B7 09                    Q22:    MOV     BH,09H          ; SET MSB OF ERROR CODE
04A5  E9 09BC R                        JMP     E_MSG
                               ; DONE WITH TEST RESET TO 40X25 - COLOR
                                       ASSUME  DS:DATA
04A8  E8 138B R                Q23:    CALL    DDS
04AB  B8 0001                          MOV     AX,0001H        ; INIT TO 40X25 - COLOR
04AE  CD 10                            INT     10H
04B0  B8 0507                          MOV     AX,0507H        ; SET TO VIDEO PAGE 7
04B3  CD 10                            INT     10H
04B5  81 3E 0072 R 1234                CMP     RESET_FLAG,1234H ; WARM START?
04BB  74 03                            JE      Q24             ; BYPASS PUTTING UP POWER-ON SCREEN
04BD  E8 0C21 R                        CALL    PUT_LOGO        ; PUT LOGO ON SCREEN
```

```
04BD  E8 0C21 R                        CALL    PUT_LOGO        ; PUT LOGO ON SCREEN
04C0  B0 76                    Q24:    MOV     AL,01110110B    ; RE-INIT TIMER 1
04C2  E6 43                            OUT     TIM_CTL,AL      ;
04C4  B0 00                            MOV     AL,00H
04C6  E6 41                            OUT     TIMER+1,AL
04C8  90                               NOP
04C9  90                               NOP
04CA  E6 41                            OUT     TIMER+1,AL
                                       ASSUME  DS:ABS0
04CC  E8 E6D8 R                        CALL    MFG_UP          ; MFG CHECKPOINT=F4
04CF  33 C0                            XOR     AX,AX
04D1  8E D8                            MOV     DS,AX
04D3  C7 06 0008 R 0F78 R              MOV     NMI_PTR,OFFSET KBDNMI ; SET INTERRUPT VECTOR
04D9  C7 06 0120 R F068 R              MOV     KEY62_PTR,OFFSET KEY_SCAN_SAVE  ; SET VECTOR FOR
                                                               ; POD INT HANDLER
04DF  0E                               PUSH    CS
04E0  58                               POP     AX
04E1  A3 0122 R                        MOV     KEY62_PTR+2,AX
                                       ASSUME  DS:DATA
04E4  E8 1388 R                        CALL    DDS             ; SET DATA SEGMENT
04E7  BE 001E R                        MOV     SI,OFFSET KB_BUFFER ; SET KEYBOARD PARMS
04EA  89 36 001A R                     MOV     BUFFER_HEAD,SI
04EE  89 36 001C R                     MOV     BUFFER_TAIL,SI
04F2  89 36 0080 R                     MOV     BUFFER_START,SI
04F6  83 C6 20                         ADD     SI,32           ; SET DEFAULT BUFFER OF 32 BYTES
04F9  89 36 0082 R                     MOV     BUFFER_END,SI
04FD  E4 A0                            IN      AL,0A0H         ; CLEAR NMI F/F
04FF  B0 80                            MOV     AL,80H          ; ENABLE NMI
0501  E6 A0                            OUT     0A0H,AL         ;
                               ;  IF A KEY IS STUCK, THE BUFFER SHOULD FILL WITH THAT KEY'S CODE
                               ;  THIS WILL BE CHECKED LATER
                               ;-----------------------------------------------
                               ;          MEMORY SIZE DETERMINE AND TEST
                               ;  THIS ROUTINE WILL DETERMINE HOW MUCH MEM
                               ;  IS ATTACHED TO THE SYSTEM (UP TO 640KB)
                               ;  AND SET "MEMORY_SIZE" AND "REAL_MEMORY"
                               ;  WORDS IN THE DATA AREA.
                               ;
                               ;  AFTER THIS, MEMORY WILL BE EITHER TESTED
                               ;  OR CLEARED, DEPENDING ON THE CONTENTS OF
                               ;  "RESET_FLAG".
                               ;   MFG. ERROR CODES   -0AXX PLANAR BD ERROR
                               ;                      -0BXX 64K CD ERROR
                               ;                      -0CXX ERRORS IN BOTH
                               ;                       ODD AND EVEN BYTES
                               ;                       IN A 128K SYS
                               ;                      -1YXX MEMORY ABOVE 128K
                               ;                       Y=SEGMENT HAVING TROUBLE
                               ;                       XX= ERROR BITS
                               ;-----------------------------------------------
                                       ASSUME  DS:DATA
0503  E8 E6D8 R                        CALL    MFG_UP          ; MFG CHECKPOINT=F3
0506  BB 0040                          MOV     BX,64           ; START WITH BASE 64K
0509  E4 62                            IN      AL,PORT_C       ; GET CONFIG BYTE
050B  A8 08                            TEST    AL,00001000B    ; SEE IF 64K CARD INSTALLED
050D  75 03                            JNE     Q25             ; (BIT 4 WILL BE 0 IF CARD PLUGGED)
050F  83 C3 40                         ADD     BX,64           ; ADD 64K
0512  53                       Q25:    PUSH    BX              ; SAVE K COUNT
0513  83 EB 10                         SUB     BX,16           ; SUBTRACT 16K CRT REFRESH SPACE
0516  89 1E 0013 R                     MOV     [MEMORY_SIZE],BX ; LOAD "CONTIGUOUS MEMORY" WORD
051A  5B                               POP     BX
051B  BA 2000                          MOV     DX,2000H        ; SET POINTER TO JUST ABOVE 128K
051E  2B FF                            SUB     DI,DI           ; SET DI TO POINT TO BEGINNING
0520  B9 AA55                          MOV     CX,0AA55H       ; LOAD DATA PATTERN
0523  8E C2                    Q26:    MOV     ES,DX           ; SET SEGMENT TO POINT TO MEMORY
                                                               ; SPACE
0525  26: 89 0D                        MOV     ES:[DI],CX      ; SET DATA PATTERN TO MEMORY
0528  B0 0F                            MOV     AL,0FH          ; SET AL TO ODD VALUE
052A  26: 8B 05                        MOV     AX,ES:[DI]      ; GET DATA PATTERN BACK FROM MEM
052D  33 C1                            XOR     AX,CX           ; SEE IF DATA MADE IT BACK
052F  75 0C                            JNZ     Q27             ; NO? THEN END OF MEM HAS BEEN
                                                               ; REACHED
0531  81 C2 1000                       ADD     DX,1000H        ; POINT TO BEGINNING OF NEXT 64K
0535  83 C3 40                         ADD     BX,64           ; ADJUST TOTAL MEM. COUNTER
0538  80 FE A0                         CMP     DH,0A0H         ; PAST 640K YET?
053B  75 E6                            JNE     Q26             ; CHECK FOR ANOTHER BLOCK IF NOT
053D  89 1E 0015 R             Q27:    MOV     [TRUE_MEM],BX   ; LOAD "TOTAL MEMORY" WORD
                               ; SIZE HAS BEEN DETERMINED, NOW TEST OR CLEAR ALL OF MEMORY
0541  B8 0004                          MOV     AX,4            ; 4 KB KNOWN OK AT THIS POINT
0544  E8 05BC R                        CALL    Q35
0547  BA 0080                          MOV     DX,0080H        ; SET POINTER TO JUST ABOVE
                                                               ; LOWER 2K
054A  B9 7800                          MOV     CX,7800H        ; TEST 30K WORDS (60KB)
054D  8E C2                    Q28:    MOV     ES,DX
054F  51                               PUSH    CX              ;
0550  53                               PUSH    BX
0551  50                               PUSH    AX
0552  E8 0859 R                        CALL    PODSTG          ; TEST OR FILL MEM
0555  74 03                            JZ      Q29
0557  E9 0603 R                        JMP     Q39             ; JUMP IF ERROR
055A  58                       Q29:    POP     AX
055B  5B                               POP     BX
055C  59                               POP     CX              ; RECOVER
055D  80 FD 78                         CMP     CH,78H          ; WAS THIS A 60 K PASS
0560  9C                               PUSHF
0561  05 003C                          ADD     AX,60           ; BUMP GOOD STORAGE BY 60 KB
0564  9D                               POPF
0565  74 03                            JE      Q30
0567  05 0002                          ADD     AX,2            ;ADD 2 FOR A 62K PASS
056A  E8 05BC R                Q30:    CALL    Q35
056D  3B C3                            CMP     AX,BX           ; ARE WE DONE YET?
056F  75 03                            JNE     Q31
0571  E9 0640 R                        JMP     Q43             ; ALL DONE, IF SO
```

```
0574  3D 0080                 Q31:      CMP     AX,128             ; DONE WITH 1ST 128K?
0577  74 1E                             JE      Q32                ; GO FINISH REST OF MEM.
0579  BA 0F80                           MOV     DX,0F80H           ; SET POINTER TO FINISH 1ST 64 KB
057C  B9 0400                           MOV     CX,0400H
057F  8E C2                             MOV     ES,DX
0581  50                                PUSH    AX
0582  53                                PUSH    BX
0583  52                                PUSH    DX
0584  E8 0B59 R                         CALL    PODSTG             ; GO TEST/FILL
0587  75 7A                             JNZ     Q39                ;
0589  5A                                POP     DX
058A  5B                                POP     BX
058B  58                                POP     AX
058C  05 0002                           ADD     AX,2               ; UPDATE GOOD COUNT
058F  BA 1000                           MOV     DX,1000H           ; SET POINTER TO 2ND 64K BLOCK
0592  B9 7C00                           MOV     CX,7C00H           ; 62K WORTH
0595  EB B6                             JMP     Q28                ; GO TEST IT
0597  BA 2000                 Q32:      MOV     DX,2000H           ; POINT TO BLOCK ABOVE 128K
059A  3B D8                   Q33:      CMP     BX,AX              ; COMPARE GOOD MEM TO TOTAL MEM
059C  75 03                             JNE     Q34
059E  E9 0640 R                         JMP     Q43                ; EXIT IF ALL DONE
05A1  B9 4000                 Q34:      MOV     CX,4000H           ; SET FOR 32KB BLOCK
05A4  8E C2                             MOV     ES,DX
05A6  50                                PUSH    AX
05A7  53                                PUSH    BX
05A8  52                                PUSH    DX
05A9  E8 0B59 R                         CALL    PODSTG             ; GO TEST/FILL
05AC  75 55                             JNZ     Q39                ;
05AE  5A                                POP     DX
05AF  5B                                POP     BX
05B0  58                                POP     AX
05B1  05 0020                           ADD     AX,32              ; BUMP GOOD MEMORY COUNT
05B4  E8 05BC R                         CALL    Q35                ; DISPLAY CURRENT GOOD MEM
05B7  80 C6 08                          ADD     DH,08H             ; SET POINTER TO NEXT 32K
05BA  EB DE                             JMP     Q33                ; AND MAKE ANOTHER PASS
                              ;---------------------------------------
                              ;     SUBROUTINE FOR PRINTING TESTED
                              ;     MEMORY OK MSG ON THE CRT
                              ; CALL PARMS: AX = K OF GOOD MEMORY
                              ;             (IN HEX)
                              ;---------------------------------------
05BC                          Q35       PROC    NEAR
05BC  E8 138B R                         CALL    DDS                ; ESTABLISH ADDRESSING
05BF  81 3E 0072 R 1234                 CMP     RESET_FLAG,1234H ; WARM START?
05C5  74 3B                             JE      Q35E               ; NO PRINT ON WARM START
05C7  53                                PUSH    BX
05C8  51                                PUSH    CX
05C9  52                                PUSH    DX
05CA  50                                PUSH    AX                 ; SAVE WORK REGS
05CB  B4 02                             MOV     AH,2               ; SET CURSOR TOWARD THE END OF
05CD  BA 1421                           MOV     DX,1421H           ; ROW 20 (ROW 20, COL. 33)
05D0  B7 07                             MOV     BH,7               ; PAGE 7
05D2  CD 10                             INT     10H
05D4  58                                POP     AX                 ;
05D5  50                                PUSH    AX
05D6  BB 000A                           MOV     BX,10              ; SET UP FOR DECIMAL CONVERT
05D9  B9 0003                           MOV     CX,3               ; OF 3 NIBBLES
05DC  33 D2                   Q36:      XOR     DX,DX              ;
05DE  F7 F3                             DIV     BX                 ; DEVIDE BY 10
05E0  80 CA 30                          OR      DL,30H             ; MAKE INTO ASCII
05E3  52                                PUSH    DX                 ; SAVE
05E4  E2 F6                             LOOP    Q36                ;
05E6  B9 0003                           MOV     CX,3               ;
05E9  58                      Q37:      POP     AX                 ; RECOVER A NUMBER
05EA  E8 18BA R                         CALL    PRT_HEX
05ED  E2 FA                             LOOP    Q37
05EF  B9 0003                           MOV     CX,3
05F2  BE 0025 R                         MOV     SI,OFFSET F3B    ; PRINT " KB"
05F5  2E: 8A 04               Q38:      MOV     AL,CS:[SI]
05F8  46                                INC     SI
05F9  E8 18BA R                         CALL    PRT_HEX
05FC  E2 F7                             LOOP    Q38
05FE  58                                POP     AX
05FF  5A                                POP     DX
0600  59                                POP     CX
0601  5B                                POP     BX
0602  C3                      Q35E:     RET
0603                          Q35       ENDP
                              ; ON ENTRY TO MEMORY ERROR ROUTINE, CX HAS ERROR BITS
                              ; AH HAS ODD/EVEN INFO, OTHER USEFUL INFO ON THE STACK
0603  5A                      Q39:      POP     DX                 ; POP SEGMENT POINTER TO DX
                                                                   ; (HEADING DOWNHILL, DON'T CARE
                                                                   ; ABOUT STACK)
0604  81 FA 2000                        CMP     DX,2000H           ; ABOVE 128K (THE SIMPLE CASE)
0608  7C 0E                             JL      Q40                ; GO DO ODD/EVEN-LESS THAN 128K
060A  8A D9                             MOV     BL,CL              ; FORM ERROR BITS ("XX")
060C  0A DD                             OR      BL,CH
060E  B1 04                             MOV     CL,4               ; ROTATE MOST SIGNIFIGANT
                                                                   ; NIBBLE OF SEGMET
0610  D2 EE                             SHR     DH,CL              ; TO LOW NIBBLE OF DH
0612  B7 10                             MOV     BH,10H             ;
0614  0A FE                             OR      BH,DH              ; FORM "1Y" VALUE
0616  EB 20                             JMP     SHORT Q42
0618  B7 0A                   Q40:      MOV     BH,0AH             ; ERROR 0A....
061A  E4 62                             IN      AL,PORT_C          ; GET CONFIG BITS
061C  24 08                             AND     AL,00001000B       ; TEST FOR ATTRIB CARD PRESENT
061E  74 06                             JZ      Q41                ; WORRY ABOUT ODD/EVEN IF IT IS
0620  8A D9                             MOV     BL,CL
0622  0A DD                             OR      BL,CH              ; COMBINE ERROR BITS IF IT ISN'T
0624  EB 12                             JMP     SHORT Q42          ;
```

```
0626  80 FC 02                 Q41:    CMP     AH,02           ; EVEN BYTE ERROR? ERR 0AXX
0629  8A D9                            MOV     BL,CL
062B  74 0B                            JE      Q42
062D  FE C7                            INC     BH              ; MAKE INTO 0BXX ERR
062F  0A DD                            OR      BL,CH           ; MOVE AND COMBINE ERROR BITS
0631  80 FC 01                         CMP     AH,1            ; ODD BYTE ERROR
0634  74 02                            JE      Q42
0636  FE C7                            INC     BH              ; MUST HAVE BEEN BOTH
                                                               ; - MAKE INTO 0CXX
0638  BE 0035 R                Q42:    MOV     SI,OFFSET MEM_ERR
063B  E8 09BC R                        CALL    E_MSG           ; LET ERROR ROUTINE FIGURE OUT
                                                               ; WHAT TO DO
063E  FA                               CLI                     ;
063F  F4                               HLT
0640                           Q43:
                               ;-----------------------------------------------
                               ;       KEYBOARD TEST
                               ; DESCRIPTION
                               ;       NMI HAS BEEN ENABLED FOR QUITE A FEW
                               ;       SECONDS NOW. CHECK THAT NO SCAN CODES
                               ;       HAVE SHOWN UP IN THE BUFFER. (STUCK
                               ;       KEY) IF THEY HAVE, DISPLAY THEM AND
                               ;       POST ERROR.
                               ;       MFG ERR CODE
                               ;       2000 STRAY NMI INTERRUPTS OR KEYBOARD
                               ;       RECEIVE ERRORS
                               ;       21XX  CARD FAILURE
                               ;           XX=01, KB DATA STUCK HIGH
                               ;           XX=02, KB DATA STUCK LOW
                               ;           XX=03, NO NMI INTERRUPT
                               ;       22XX STUCK KEY (XX=SCAN CODE)
                               ;-----------------------------------------------
                                       ASSUME  DS:DATA
                               ;----- CHECK FOR STUCK KEYS
0640  E8 E6D8 R                        CALL    MFG_UP          ; MFG CODE=F2
0643  E8 138B R                        CALL    DDS             ; ESTABLISH ADDRESSING
0646  BB 001E R                        MOV     BX,OFFSET KB_BUFFER
0649  8A 07                            MOV     AL,[BX]         ; CHECK FOR STUCK KEYS
064B  0A C0                            OR      AL,AL           ; SCAN CODE = 0?
064D  74 06                            JE      F6_Y            ; YES - CONTINUE TESTING
064F  B7 22                            MOV     BH,22H          ; 22XX ERROR CODE
0651  8A D8                            MOV     BL,AL           ;
0653  EB 0A                            JMP     SHORT F6
0655  80 3E 0012 R 00          F6_Y:   CMP     KBD_ERR,00H     ; DID NMI'S HAPPEN WITH NO SCAN
                                                               ; CODE PASSED?
065A  74 1C                            JE      F7              ; (STRAYS) - CONTINUE IF NONE
065C  BB 2000                          MOV     BX,2000H        ; SET ERROR CODE 2000
065F  BE 0036 R                F6:     MOV     SI,OFFSET KEY_ERR ; GET MSG ADDR
0662  81 3E 0072 R 4321                CMP     RESET_FLAG,4321H ; WARM START TO DIAGS
0668  74 0B                            JE      F6_Z            ; DO NOT PUT UP MESSAGE
066A  81 3E 0072 R 1234                CMP     RESET_FLAG,1234H ; WARM SYSTEM START
0670  74 03                            JE      F6_Z            ; DO NOT PUT UP MESSAGE
0672  E8 09BC R                        CALL    E_MSG           ; PRINT MSG ON SCREEN
0675  E9 06FF R                F6_Z:   JMP     F6_X
                               ; CHECK LINK CARD, IF PRESENT
0678  BA 0201                  F7:     MOV     DX,0201H
067B  EC                               IN      AL,DX           ; CHECK FOR BURN-IN MODE
067C  24 F0                            AND     AL,0F0H
067E  74 7F                            JZ      F6_X            ; BYPASS CHECK IN BURN-IN MODE
0680  E4 62                            IN      AL,PORT_C       ; GET CONFIG. PORT DATA
0682  24 80                            AND     AL,10000000B    ; KEYBOARD CABLE ATTACHED?
0684  74 79                            JZ      F6_X            ; BYPASS  TEST IF IT IS
0686  E4 61                            IN      AL,PORT_B       ;
0688  24 FC                            AND     AL,11111100B    ; DROP SPEAKER DATA
068A  E6 61                            OUT     PORT_B,AL       ;
068C  B0 B6                            MOV     AL,0B6H         ; MODE SET TIMER 2
068E  E6 43                            OUT     TIM_CTL,AL      ;
0690  B0 40                            MOV     AL,040H         ; DISABLE NMI
0692  E6 A0                            OUT     0A0H,AL         ;
0694  B0 20                            MOV     AL,32           ; LSB TO TIMER 2
                                                               ; (APPROX. 40Khz VALUE)
0696  BA 0042                          MOV     DX,TIMER+2
0699  EE                               OUT     DX,AL
069A  2B C0                            SUB     AX,AX
069C  8B C8                            MOV     CX,AX
069E  EE                               OUT     DX,AL           ; MSB TO TIMER 2 (START TIMER)
069F  E4 61                            IN      AL,PORT_B       ;
06A1  0C 01                            OR      AL,1
06A3  E6 61                            OUT     PORT_B,AL       ; ENABLE TIMER 2
06A5  E4 62                    F7_0:   IN      AL,PORT_C       ; SEE IF KEYBOARD DATA ACTIVE
06A7  24 40                            AND     AL,01000000B    ;
06A9  75 06                            JNZ     F7_1            ; EXIT LOOP IF DATA SHOWED UP
06AB  E2 F8                            LOOP    F7_0
06AD  B3 02                            MOV     BL,02H          ; SET NO KEYBOARD DATA ERROR
06AF  EB 49                            JMP     SHORT F6_1
06B1  06                       F7_1:   PUSH    ES              ; SAVE ES
06B2  2B C0                            SUB     AX,AX           ; SET UP SEGMENT REG
06B4  8E C0                            MOV     ES,AX           ; *
06B6  26: C7 06 0008 R F815 R          MOV     ES:[NMI_PTR],OFFSET D11 ; SET UP NEW NMI VECTOR
06BD  A2 0084 R                        MOV     INTR_FLAG,AL    ; RESET INTR FLAG
06C0  E4 61                            IN      AL,PORT_B       ; DISABLE INTERNAL BEEPER TO
06C2  0C 30                            OR      AL,00110000B    ; PREVENT ERROR BEEP
06C4  E6 61                            OUT     PORT_B,AL
06C6  B0 C0                            MOV     AL,0C0H
06C8  E6 A0                            OUT     0A0H,AL         ; ENABLE NMI
06CA  B9 0100                          MOV     CX,0100H        ;
```

```
06CD  E2 FE                       F6_O:     LOOP     F6_O              ; WAIT A BIT
06CF  E4 61                                 IN       AL,PORT_B         ; RE-ENABLE BEEPER
06D1  24 CF                                 AND      AL,11001111B
06D3  E6 61                                 OUT      PORT_B,AL
06D5  A0 0084 R                             MOV      AL,INTR_FLAG      ; GET INTR FLAG
06D8  0A C0                                 OR       AL,AL             ; WILL BE NON-ZERO IF NMI HAPPENED
06DA  B3 03                                 MOV      BL,03H            ; SET POSSIBLE ERROR CODE
06DC  26: C7 06 0008 R 0F78 R               MOV      ES:[NMI_PTR],OFFSET KBDNMI ; RESET NMI VECTOR
06E3  07                                    POP      ES                ; RESTORE ES
06E4  74 14                                 JZ       F6_1              ; JUMP IF NO NMI
06E6  B0 00                                 MOV      AL,00H            ; DISABLE FEEDBACK CKT
06E8  E6 A0                                 OUT      0A0H,AL           ;
06EA  E4 61                                 IN       AL,PORT_B         ;
06EC  24 FE                                 AND      AL,11111110B      ; DROP GATE TO TIMER 2
06EE  E6 61                                 OUT      PORT_B,AL         ;
06F0  E4 62                       F6_2:     IN       AL,PORT_C         ; SEE IF KEYBOARD DATA ACTIVE
06F2  24 40                                 AND      AL,01000000B      ;
06F4  74 09                                 JZ       F6_X              ; EXIT LOOP IF DATA WENT LOW
06F6  E2 F8                                 LOOP     F6_2
06F8  B3 01                                 MOV      BL,01H            ; SET KEYBOARD DATA STUCK HIGH ERR
06FA  B7 21                       F6_1:     MOV      BH,21H            ; POST ERROR  "21XX"
06FC  E9 065F R                             JMP      F6                ;
06FF  B0 00                       F6_X:     MOV      AL,00H            ; DISABLE FEEDBACK CKT
0701  E6 A0                                 OUT      0A0H,AL           ;
                                  ;--------------------------------------------------------
                                  ;         CASSETTE INTERFACE TEST
                                  ; DESCRIPTION
                                  ;         TURN CASSETTE MOTOR OFF. WRITE A BIT OUT TO THE
                                  ;         CASSETTE DATA BUS. VERIFY THAT CASSETTE DATA
                                  ;         READ IS WITHIN A VALID RANGE.
                                  ;   MFG. ERROR CODE=2300H (DATA PATH ERROR)
                                  ;                   23FF (RELAY FAILED TO PICK)
                                  ;--------------------------------------------------------
= 0A9A                            MAX_PERIOD       EQU      0A9AH  ;NOM.+10%
= 08AD                            MIN_PERIOD       EQU      08ADH  ;NOM -10%
                                  ;----- TURN THE CASSETTE MOTOR OFF
0703  E8 E6D8 R                             CALL     MFG_UP            ; MFG CODE=F1
0706  E4 61                                 IN       AL,PORT_B
0708  0C 09                                 OR       AL,00001001B      ; SET TIMER 2 SPK OUT, AND CASSETTE
070A  E6 61                                 OUT      PORT_B,AL         ; OUT BITS ON, CASSETTE MOT OFF
                                  ;----- WRITE A BIT
070C  E4 21                                 IN       AL,INTA01
070E  0C 01                                 OR       AL,01H            ; DISABLE TIMER INTERRUPTS
0710  E6 21                                 OUT      INTA01,AL
0712  B0 B6                                 MOV      AL,0B6H           ; SEL TIM 2, LSB, MSB, MD 3
0714  E6 43                                 OUT      TIMER+3,AL        ; WRITE 8253 CMD/MODE REG
0716  B8 04D2                               MOV      AX,1234           ; SET TIMER 2 CNT FOR 1000 USEC
0719  E6 42                                 OUT      TIMER+2,AL        ; WRITE TIMER 2 COUNTER REG
071B  8A C4                                 MOV      AL,AH             ; WRITE MSB
071D  E6 42                                 OUT      TIMER+2,AL
071F  2B C9                                 SUB      CX,CX             ; CLEAR COUNTER FOR LONG DELAY
0721  E2 FE                                 LOOP     $                 ; WAIT FOR COUNTER TO INIT
                                  ;----- READ CASSETTE INPUT
0723  E4 62                                 IN       AL,PORT_C         ; READ VALUE OF CASS IN BIT
0725  24 10                                 AND      AL,10H            ; ISOLATE FROM OTHER BITS
0727  A2 006B R                             MOV      LAST_VAL,AL
072A  E8 F96F R                             CALL     READ_HALF_BIT     ; TO SET UP CONDITIONS FOR CHECK
072D  E8 F96F R                             CALL     READ_HALF_BIT
0730  E3 3E                                 JCXZ     F8                ; CAS_ERR
0732  53                                    PUSH     BX                ; SAVE HALF BIT TIME VALUE
0733  E8 F96F R                             CALL     READ_HALF_BIT
0736  58                                    POP      AX                ; GET TOTAL TIME
0737  E3 37                                 JCXZ     F8                ; CAS_ERR
0739  03 C3                                 ADD      AX,BX
073B  3D 0A9A                               CMP      AX,MAX_PERIOD
073E  73 30                                 JNC      F8                ; CAS_ERR
0740  3D 08AD                               CMP      AX,MIN_PERIOD
0743  72 2B                                 JC       F8
0745  BA 0201                               MOV      DX,201H
0748  EC                                    IN       AL,DX
0749  24 F0                                 AND      AL,0F0H           ; DETERMINE MODE
074B  3C 10                                 CMP      AL,00010000B      ; MFG?
074D  74 04                                 JE       F9
074F  3C 40                                 CMP      AL,01000000B      ; SERVICE?
0751  75 26                                 JNE      T13_END           ; GO TO NEXT TEST IF NOT
                                  ; CHECK THAT CASSETTE RELAY IS PICKING (CAN'T DO TEST IN NORMAL
                                  ; MODE BECAUSE OF POSSIBILITY OF WRITING ON CASSETTE IF "RECORD"
                                  ; BUTTON IS DEPRESSED.)
0753  E4 61                       F9:       IN       AL,PORT_B
0755  8A D0                                 MOV      DL,AL             ; SAVE PORT B CONTENTS
0757  24 E5                                 AND      AL,11100101B      ; SET CASSETTE MOTOR ON
0759  E6 61                                 OUT      PORT_B,AL         ;
075B  33 C9                                 XOR      CX,CX
075D  E2 FE                       F91:      LOOP     F91               ; WAIT FOR RELAY TO SETTLE
075F  E8 F96F R                             CALL     READ_HALF_BIT
0762  E8 F96F R                             CALL     READ_HALF_BIT
0765  8A C2                                 MOV      AL,DL             ; DROP RELAY
0767  E6 61                                 OUT      PORT_B,AL
0769  E3 0E                                 JCXZ     T13_END           ; READ_HALF_BIT SHOULD TIME OUT IN
                                                                       ; THIS SITUATION
076B  BB 23FF                               MOV      BX,23FFH          ; ERROR 23FF
076E  EB 03                                 JMP      SHORT F81
0770                              F8:                                  ; CAS_ERR
0770  BB 2300                               MOV      BX,2300H          ; ERR. CODE 2300H
0773  BE 0037 R                   F81:      MOV      SI,OFFSET CASS_ERR ; CASSETTE WRAP FAILED
0776  E8 09BC R                             CALL     E_MSG             ; GO PRINT ERROR MSG
0779  E4 21                       T13_END:IN         AL,INTA01
077B  24 FE                                 AND      AL,0FEH           ; ENABLE TIMER INTS
077D  E6 21                                 OUT      INTA01,AL
077F  E4 A0                                 IN       AL,NMI_PORT       ; CLEAR NMI FLIP/FLOP
0781  B0 80                                 MOV      AL,80H            ; ENABLE NMI INTERUPTS
0783  E6 A0                                 OUT      NMI_PORT,AL
```

```
                                       ;-----------------------------------------------------------------
                                       ;       SERIAL PRINTER AND MODEM POWER ON DIAGNOSTIC
                                       ;   DESCRIPTION:
                                       ;       VERIFIES THAT THE SERIAL PRINTER UART FUNCTIONS PROPERLY.
                                       ;       CHECKS IF THE MODEM CARD IS ATTACHED.  IF IT'S NOT, EXITS.
                                       ;       VERIFIES THAT THE MODEM UART FUNCTIONS PROPERLY.
                                       ;       ERROR CODES RETURNED BY 'UART' RANGE FROM 1 TO 1FH AND ARE
                                       ;       REPORTED VIA REGISTER BL.  SEE LISTING OF 'UART' (POD27)
                                       ;       FOR POSSIBLE ERRORS.
                                       ;       MFG. ERR. CODES  23XX FOR SERIAL PRINTER
                                       ;                        24XX FOR MODEM
                                       ;-----------------------------------------------------------------
                                               ASSUME  CS:CODE,DS:DATA
                                       ;-----------------------------------------------------------------
                                       ;       TEST SERIAL PRINTER INS8250 UART
                                       ;-----------------------------------------------------------------
0785  E8 E6D8 R                                CALL    MFG_UP           ; MFG ROUTINE INDICATOR=F0
0788  BA 02F8                                  MOV     DX,02F8H         ; ADDRESS OF SERIAL PRINTER CARD
078B  E8 E831 R                                CALL    UART             ; ASYNCH. COMM. ADAPTER POD
078E  73 06                                    JNC     TM               ; PASSED
0790  BE 0038 R                                MOV     SI,OFFSET COM1_ERR ; CODE FOR DISPLAY
0793  E8 09BC R                                CALL    E_MSG            ; REPORT ERROR
                                       ;-----------------------------------------------------------------
                                       ;       TEST MODEM INS8250 UART
                                       ;-----------------------------------------------------------------
0796  E8 E6D8 R                        TM:     CALL    MFG_UP           ; MFG ROUTINE INDICATOR = EF
0799  E4 62                                    IN      AL,PORT_C        ; TEST FOR MODEM CARD PRESENT
079B  24 02                                    AND     AL,00000010B     ; ONLY CONCERNED WITH BIT 1
079D  75 0E                                    JNE     TM1              ; IT'S NOT THERE - DONE WITH TEST
079F  BA 03F8                                  MOV     DX,03F8H         ; ADDRESS OF MODEM CARD
07A2  E8 E831 R                                CALL    UART             ; ASYNCH. COMM. ADAPTER POD
07A5  73 06                                    JNC     TM1              ; PASSED
07A7  BE 0039 R                                MOV     SI,OFFSET COM2_ERR ; MODEM ERROR
07AA  E8 09BC R                                CALL    E_MSG            ; REPORT ERROR
07AD                                   TM1:
                                       ;----------------------------------------------
                                       ;       SETUP HARDWARE INT. VECTOR TABLE
                                       ;----------------------------------------------
                                               ASSUME  CS:CODE,DS:ABS0
07AD  2B C0                                    SUB     AX,AX
07AF  8E C0                                    MOV     ES,AX
07B1  B9 0008                                  MOV     CX,08            ; GET VECTOR CNT
07B4  0E                                       PUSH    CS               ; SETUP DS SEG REG
07B5  1F                                       POP     DS
07B6  BE FEF3 R                                MOV     SI,OFFSET VECTOR_TABLE
07B9  BF 0020 R                                MOV     DI,OFFSET INT_PTR
07BC  A5                               F7A:    MOVSW
07BD  47                                       INC     DI               ; SKIP OVER SEGMENT
07BE  47                                       INC     DI
07BF  E2 FB                                    LOOP    F7A
                                       ;----- SET UP OTHER INTERRUPTS AS NECESSARY
                                               ASSUME  DS:ABS0
07C1  8E D9                                    MOV     DS,CX
07C3  C7 06 0014 R FF54 R                      MOV     INT5_PTR,OFFSET PRINT_SCREEN ; PRINT SCREEN
07C9  C7 06 0120 R 10C6 R                      MOV     KEY62_PTR,OFFSET KEY62_INT ; 62 KEY CONVERSION
                                                                                  ; ROUTINE
07CF  C7 06 0110 R FA6E R                      MOV     CSET_PTR,OFFSET CRT_CHAR_GEN ; DOT TABLE
07D5  C7 06 0060 R FFCB R                      MOV     BASIC_PTR,OFFSET BAS_ENT ; CASSETTE BASIC ENTRY
07DB  0E                                       PUSH    CS
07DC  58                                       POP     AX
07DD  A3 0062 R                                MOV     WORD PTR BASIC_PTR+2,AX ; CODE SEGMENT FOR CASSETTE
                                       ;---------------------------------------------------------------
                                       ; CHECK FOR OPTIONAL ROM FROM C0000 TO F0000 IN 2K BLOCKS
                                       ;       (A VALID MODULE HAS '55AA' IN THE FIRST 2 LOCATIONS,
                                       ;       LENGTH INDICATOR (LENGTH/512) IN THE 3D LOCATION AND
                                       ;       TEST/INIT. CODE STARTING IN THE 4TH LOCATION.)
                                       ;       MFG ERR CODE 25XX (XX=MSB OF SEGMENT THAT HAS CRC CHECK)
                                       ;---------------------------------------------------------------
07E0  B0 01                                    MOV     AL,01H
07E2  E6 13                                    OUT     13H,AL
07E4  E8 E6D8 R                                CALL    MFG_UP           ; MFG ROUTINE = EE
07E7  BA C000                                  MOV     DX,0C000H        ; SET BEGINNING ADDRESS
07EA                                   ROM_SCAN_1:
07EA  8E DA                                    MOV     DS,DX
07EC  2B DB                                    SUB     BX,BX            ; SET BX=0000
07EE  8B 07                                    MOV     AX,[BX]          ; GET 1ST WORD FROM MODULE
07F0  53                                       PUSH    BX
07F1  5B                                       POP     BX               ; BUS SETTLING
07F2  3D AA55                                  CMP     AX,0AA55H        ; = TO ID WORD?
07F5  75 05                                    JNZ     NEXT_ROM         ; PROCEED TO NEXT ROM IF NOT
07F7  E8 EB51 R                                CALL    ROM_CHECK        ; GO CHECK OUT MODULE
07FA  EB 04                                    JMP     SHORT ARE_WE_DONE ; CHECK FOR END OF ROM SPACE
07FC                                   NEXT_ROM:
07FC  81 C2 0080                               ADD     DX,0080H         ; POINT TO NEXT 2K ADDRESS
0800                                   ARE_WE_DONE:
0800  81 FA F000                               CMP     DX,0F000H        ; AT F0000 YET?
0804  7C E4                                    JL      ROM_SCAN_1       ; GO CHECK ANOTHER ADD. IF NOT
```

```
                               ;--------------------------------------------------------
                               ;          DISKETTE ATTACHMENT TEST
                               ; DESCRIPTION
                               ;          CHECK IF IPL DISKETTE DRIVE IS ATTACHED TO SYSTEM.  IF
                               ;          ATTACHED, VERIFY STATUS OF NEC FDC AFTER A RESET. ISSUE
                               ;          A RECAL AND SEEK CMD TO FDC AND CHECK STATUS. COMPLETE
                               ;          SYSTEM INITIALIZATION THEN PASS CONTROL TO THE BOOT
                               ;          LOADER PROGRAM.
                               ;    MFG ERR CODES: 2601 RESET TO DISKETTE CONTROLLER CD. FAILED
                               ;                   2602 RECALIBRATE TO DISKETTE DRIVE FAILED
                               ;                   2603 WATCHDOG TIMER FAILED
                               ;--------------------------------------------------------
                                          ASSUME  CS:CODE,DS:DATA
0806  E8 E6D8 R                           CALL    MFG_UP            ; MFG ROUTINE = ED
0809  E8 138B R                           CALL    DDS               ; POINT TO DATA AREA
080C  B0 FF                               MOV     AL,0FFH
080E  A2 0074 R                           MOV     TRACK0,AL         ; INIT DISKETTE SCRATCHPADS
0811  A2 0075 R                           MOV     TRACK1,AL
0814  A2 0076 R                           MOV     TRACK2,AL
0817  E4 62                               IN      AL,PORT_C         ; DISKETTE PRESENT?
0819  24 04                               AND     AL,00000100B
081B  74 03                               JZ      F10_0
081D  E9 08A3 R                           JMP     F15               ; NO - BYPASS DISKETTE TEST
0820  80 0E 0010 R 01          F10_0:     OR      BYTE PTR EQUIP_FLAG,01H ; SET IPL DISKETTE
                                                                    ; INDICATOR IN EQUIP. FLAG
0825  83 3E 0072 R 00                     CMP     RESET_FLAG,0      ; RUNNING FROM POWER-ON STATE?
082A  75 0E                               JNE     F10               ; BYPASS WATCHDOG TEST
082C  B0 0A                               MOV     AL,00001010B      ; READ INT. REQUEST REGISTER CMD
082E  E6 20                               OUT     INTA00,AL         ;
0830  E4 20                               IN      AL,INTA00
0832  24 40                               AND     AL,01000000B      ; HAS WATCHDOG GONE OFF?
0834  75 04                               JNZ     F10               ; PROCEED IF IT HAS
0836  B3 03                               MOV     BL,03H            ; SET ERROR CODE
0838  EB 33                               JMP     SHORT F13
083A  B0 80                    F10:       MOV     AL,FDC_RESET
083C  E6 F2                               OUT     0F2H,AL           ; DISABLE WATCHDOG TIMER
083E  B4 00                               MOV     AH,0              ; RESET NEC FDC
0840  8A D4                               MOV     DL,AH             ; SET FOR DRIVE 0
0842  CD 13                               INT     13H               ; VERIFY STATUS AFTER RESET
0844  F6 C4 FF                            TEST    AH,0FFH           ; STATUS OK?
0847  B3 01                               MOV     BL,01H            ; SET UP POSSIBLE ERROR CODE
0849  75 22                               JNZ     F13               ; NO - FDC FAILED
                               ;----- TURN DRIVE 0 MOTOR ON
084B  B0 81                               MOV     AL,DRIVE_ENABLE+FDC_RESET ; TURN MOTOR ON,DRIVE 0
084D  E6 F2                               OUT     0F2H,AL           ; WRITE FDC CONTROL REG
084F  2B C9                               SUB     CX,CX
0851  E2 FE                    F11:       LOOP    F11               ; WAIT FOR 1 SECOND
0853  E2 FE                    F12:       LOOP    F12
0855  33 D2                               XOR     DX,DX             ; SELECT DRIVE 0
0857  B5 01                               MOV     CH,1              ; SELECT TRACK 1
0859  88 16 003E R                        MOV     SEEK_STATUS,DL
085D  E8 E9FB R                           CALL    SEEK              ; RECALIBRATE DISKETTE
0860  B3 02                               MOV     BL,02H            ; ERROR CODE
0862  72 09                               JC      F13               ; GO TO ERR SUBROUTINE IF ERR
0864  B5 22                               MOV     CH,34             ; SELECT TRACK 34
0866  E8 E9FB R                           CALL    SEEK              ; SEEK TO TRACK 34
0869  73 0A                               JNC     F14               ; OK, TURN MOTOR OFF
086B  B3 02                               MOV     BL,02H
086D  B7 26                    F13:       MOV     BH,26H            ; DSK_ERR:(26XX)
086F  BE 003C R                           MOV     SI,OFFSET DISK_ERR ; GET ADDR OF MSG
0872  E8 09BC R                           CALL    E_MSG             ; GO PRINT ERROR MSG
0875  B0 82                    F14:       MOV     AL,FDC_RESET+02H
0877  E6 F2                               OUT     0F2H,AL
0879  E4 E2                               IN      AL,0E2H
087B  24 06                               AND     AL,00000110B
087D  3C 02                               CMP     AL,00000010B
087F  75 1E                               JNE     F14_1
0881  B0 84                               MOV     AL,FDC_RESET+04H
0883  E6 F2                               OUT     0F2H,AL
0885  E4 E2                               IN      AL,0E2H
0887  24 06                               AND     AL,00000110B
0889  3C 04                               CMP     AL,00000100B
088B  75 12                               JNE     F14_1
088D  E4 E2                               IN      AL,0E2H
088F  24 30                               AND     AL,00110000B
0891  74 0C                               JZ      F14_1
0893  3C 10                               CMP     AL,00010000B
0895  B4 40                               MOV     AH,01000000B
0897  74 02                               JE      F14_2
0899  B4 80                               MOV     AH,10000000B
089B  08 26 0010 R             F14_2:     OR      BYTE PTR EQUIP_FLAG,AH
                               ;----- TURN DRIVE 0 MOTOR OFF
089F  B0 80                    F14_1:     MOV     AL,FDC_RESET      ; TURN DRIVE 0 MOTOR OFF
08A1  E6 F2                               OUT     0F2H,AL
08A3  C6 06 0084 R 00          F15:       MOV     INTR_FLAG,00H     ; SET STRAY INTERRUPT FLAG = 00
08A8  BF 0078 R                           MOV     DI,OFFSET PRINT_TIM_OUT ;SET DEFAULT PRT TIMEOUT
08AB  1E                                  PUSH    DS
08AC  07                                  POP     ES
08AD  B8 1414                             MOV     AX,1414H          ; DEFAULT=20
08B0  AB                                  STOSW
08B1  AB                                  STOSW
08B2  B8 0101                             MOV     AX,0101H          ;RS232 DEFAULT=01
08B5  AB                                  STOSW
08B6  AB                                  STOSW
08B7  E4 21                               IN      AL,INTA01
08B9  24 FE                               AND     AL,0FEH           ; ENABLE TIMER INT. (LVL 0)
08BB  E6 21                               OUT     INTA01,AL
                                          ASSUME  DS:XXDATA
08BD  1E                                  PUSH    DS
08BE  B8 ---- R                           MOV     AX,XXDATA
08C1  8E D8                               MOV     DS,AX
```

```
08C3  80 3E 0018 R 00                    CMP     POST_ERR,00H     ; CHECK FOR "POST_ERR" NON-ZERO
                                         ASSUME  DS:DATA
08C8  1F                                 POP     DS
08C9  74 10                              JE      F15A_0           ; CONTINUE IF NO ERROR
08CB  B2 02                              MOV     DL,2             ; 2 SHORT BEEPS (ERROR)
08CD  E8 1A0C R                          CALL    ERR_BEEP
08D0                             ERR_WAIT:
08D0  B4 00                              MOV     AH,00
08D2  CD 16                              INT     16H              ; WAIT FOR "ENTER" KEY
08D4  80 FC 1C                           CMP     AH,1CH
08D7  75 F7                              JNE     ERR_WAIT
08D9  EB 05                              JMP     SHORT F15C
08DB  B2 01                      F15A_0: MOV     DL,1             ; 1 SHORT BEEP (NO ERRORS)
08DD  E8 1A0C R                          CALL    ERR_BEEP
                                 ;----- SETUP PRINTER AND RS232 BASE ADDRESSES IF DEVICE ATTACHED
08E0  BD 003D R                  F15C:   MOV     BP,OFFSET F4     ; PRT_SRC_TBL
08E3  33 F6                              XOR     SI,SI
08E5                             F16:                             ; PRT_BASE:
08E5  2E: 8B 56 00                       MOV     DX,CS:[BP]       ; GET PRINTER BASE ADDR
08E9  B0 AA                              MOV     AL,0AAH          ; WRITE DATA TO PORT A
08EB  EE                                 OUT     DX,AL
08EC  1E                                 PUSH    DS               ; BUS SETTLING
08ED  EC                                 IN      AL,DX            ; READ PORT A
08EE  1F                                 POP     DS
08EF  3C AA                              CMP     AL,0AAH          ; DATA PATTERN SAME
08F1  75 06                              JNE     F17              ; NO - CHECK NEXT PRT CD
08F3  89 94 0008 R                       MOV     PRINTER_BASE[SI],DX ; YES - STORE PRT BASE ADDR
08F7  46                                 INC     SI               ; INCREMENT TO NEXT WORD
08F8  46                                 INC     SI
08F9  45                         F17:    INC     BP               ; POINT TO NEXT BASE ADDR
08FA  45                                 INC     BP
08FB  83 FD 41                           CMP     BP,OFFSET F4E    ; ALL POSSIBLE ADDRS CHECKED?
08FE  75 E5                              JNE     F16              ; PRT_BASE
0900  33 DB                              XOR     BX,BX            ; SET ADDRESS BASE
0902  BA 03FA                            MOV     DX,03FAH         ; POINT TO INT ID REGISTER
0905  EC                                 IN      AL,DX            ; READ PORT
0906  A8 F8                              TEST    AL,0F8H          ; SEEM TO BE AN 8250
0908  75 08                              JNZ     F18
090A  C7 87 0000 R 03F8                  MOV     RS232_BASE[BX],3F8H ; SETUP RS232 CD #1 ADDR
0910  43                                 INC     BX
0911  43                                 INC     BX
0912  C7 87 0000 R 02F8          F18:    MOV     RS232_BASE[BX],2F8H ; SETUP RS232 #2
0918  43                                 INC     BX               ; (ALWAYS PRESENT)
0919  43                                 INC     BX
                                 ;----- SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232
                                 ;       CARDS
091A  8B C6                              MOV     AX,SI            ; SI HAS 2* NUMBER OF PRINTERS
091C  B1 03                              MOV     CL,3             ; SHIFT COUNT
091E  D2 C8                              ROR     AL,CL            ; ROTATE RIGHT 3 POSITIONS
0920  0A C3                              OR      AL,BL            ; OR IN THE RS232 COUNT
0922  08 06 0011 R                       OR      BYTE PTR EQUIP_FLAG+1,AL ; STORE AS SECOND BYTE
                                 ;----- SET EQUIP. FLAG TO INDICATE PRESENCE OF SERIAL PRINTER
                                 ; ATTACHED TO ON BOARD RS232 PORT. ---ASSUMPTION--"RTS" IS TIED TO
                                 ; "CARRIER DETECT" IN THE CABLE PLUG FOR THIS SPECIFIC PRINTER.
0926  8B C8                              MOV     CX,AX            ; SAVE PRINTER COUNT IN CX
0928  BB 02FE                            MOV     BX,2FEH          ; SET POINTER TO MODEM STATUS REG
092B  BA 02FC                            MOV     DX,2FCH          ; POINT TO MODEM CONTROL REG
092E  2A C0                              SUB     AL,AL
0930  EE                                 OUT     DX,AL            ; CLEAR IT
0931  EB 00                              JMP     $+2              ; DELAY
0933  87 D3                              XCHG    DX,BX            ; POINT TO MODEM STATUS REG
0935  EC                                 IN      AL,DX            ; CLEAR IT
0936  EB 00                              JMP     $+2              ; DELAY
0938  B0 02                              MOV     AL,02H           ; BRING UP RTS
093A  87 D3                              XCHG    DX,BX            ; POINT TO MODEM CONTROL REG
093C  EE                                 OUT     DX,AL            ;
093D  EB 00                              JMP     $+2              ; DELAY
093F  87 D3                              XCHG    DX,BX            ; POINT TO MODEM STATUS REG
0941  EC                                 IN      AL,DX            ; GET CONTENTS
0942  A8 08                              TEST    AL,00001000B     ; HAS CARRIER DETECT CHANGED?
0944  74 23                              JZ      F19_A            ; NO, THEN NO PRINTER
0946  A8 01                              TEST    AL,00000001B     ; DID CTS CHANGE? (AS WITH WRAP
                                                                  ; CONNECTOR INSTALLED)
0948  75 1F                              JNZ     F19_A            ; WRAP CONNECTOR ON IF IT DID
094A  2A C0                              SUB     AL,AL            ; SET RTS OFF
094C  87 D3                              XCHG    DX,BX            ; POINT TO MODEM CONTROL REG
094E  EE                                 OUT     DX,AL            ; DROP RTS
094F  EB 00                              JMP     $+2              ; DELAY
0951  87 D3                              XCHG    DX,BX            ; MODEM STATUS REG
0953  EC                                 IN      AL,DX            ; GET STATUS
0954  24 08                              AND     AL,00001000B     ; HAS CARRIER DETECT CHANGED?
0956  74 11                              JZ      F19_A            ; NO, THEN NO PRINTER
                                 ; CARRIER DETECT IS FOLLOWING RTS-INDICATE SERIAL PRINTER ATTACHED
0958  80 C9 20                           OR      CL,00100000B     ;
095B  F6 C1 C0                           TEST    CL,11000000B     ; CHECK FOR NO PARALLEL PRINTERS
095E  75 09                              JNZ     F19_A            ; DO NOTHING IF PARALLEL PRINTER
                                                                  ; ATTACHED
0960  80 C9 40                           OR      CL,01000000B     ; INDICATE 1 PRINTER ATTACHED
0963  C7 06 0008 R 02F8                  MOV     PRINTER_BASE,2F8H ; STORE ON-BOARD RS232 BASE IN
                                                                  ; PRINTER BASE
0969  08 0E 0011 R               F19_A:  OR      BYTE PTR EQUIP_FLAG+1,CL ; STORE AS SECOND BYTE
096D  33 D2                              XOR     DX,DX            ; POINT TO FIRST SERIAL PORT
096F  F6 C1 40                           TEST    CL,040H          ; SERIAL PRINTER ATTACHED?
0972  74 18                              JZ      F19_C            ; NO, SKIP INIT
0974  81 3E 0000 R 02F8                  CMP     RS232_BASE,02F8H ; PRINTER IN FIRST SERIAL PORT
097A  74 01                              JE      F19_B            ; YES, JUMP
097C  42                                 INC     DX               ; NO POINT TO SECOND SERIAL PORT
097D  B8 0087                    F19_B:  MOV     AX,87H           ; INIT SERIAL PRINTER
0980  CD 14                              INT     14H
0982  F6 C4 1E                           TEST    AH,1EH           ; ERROR?
0985  75 05                              JNZ     F19_C            ; YES, JUMP
0987  B8 0118                            MOV     AX,0118H         ; SEND CANCEL COMMAND TO
098A  CD 14                              INT     14H              ; ..SERIAL PRINTER
```

```
098C  BA 0201               F19_C:  MOV     DX,0201H
098F  EC                            IN      AL,DX           ; GET MFG./ SERVICE  MODE INFO
0990  24 F0                         AND     AL,0F0H         ; IS HIGH ORDER NIBBLE = 0?
0992  75 03                         JNZ     F19_1           ; (BURN-IN MODE)
0994  E9 0043 R             F19_0:  JMP     START           ; ELSE GO TO BEGINNING OF POST
0997  3C 20                 F19_1:  CMP     AL,00100000B    ; SERVICE MODE LOOP?
0999  74 F9                         JE      F19_0           ; BRANCH TO START
099B  81 3E 0072 R 4321             CMP     RESET_FLAG,4321H ; DIAG. CONTROL PROGRAM RESTART?
09A1  74 0C                         JE      F19_3           ; NO, GO BOOT
09A3  3C 10                         CMP     AL,00010000B    ; MFG DCP RUN REQUEST
09A5  74 08                         JE      F19_3
09A7  C7 06 0072 R 1234             MOV     RESET_FLAG,1234H ; SET WARM START INDICATOR IN CASE
                                                            ; OF CARTRIDGE RESET
09AD  CD 19                         INT     19H             ; GO TO THE BOOT LOADER
                                    ASSUME  DS:ABS0
09AF  FA                    F19_3:  CLI
09B0  2B C0                         SUB     AX,AX
09B2  8E D8                         MOV     DS,AX           ; RESET TIMER INT.
09B4  C7 06 0020 R FEA5 R           MOV     INT_PTR,OFFSET TIMER_INT
09BA  CD 80                         INT     80H             ; ENTER DCP THROUGH INT. 80H
                            ;--------------------------------------------------------------
                            ;       THIS SUBROUTINE IS THE GENERAL ERROR HANDLER FOR THE POST
                            ;
                            ; ENTRY REQUIREMENTS:
                            ;       SI = OFFSET(ADDRESS) OF MESSAGE BUFFER
                            ;       BX= ERROR CODE FOR MANUFACTURING OR SERVICE MODE
                            ;       REGISTERS ARE NOT PRESERVED
                            ;       LOCATION "POST_ERR" IS SET NON-ZERO IF AN ERROR OCCURS IN
                            ;       CUSTOMER MODE
                            ;       SERVICE/MANUFACTURING FLAGS AS FOLLOWS: (HIGH NIBBLE OF
                            ;       PORT 201)
                            ;       0000 = MANUFACTURING (BURN-IN) MODE
                            ;       0001 = MANUFACTURING (SYSTEM TEST) MODE
                            ;       0010 = SERVICE MODE (LOOP POST)
                            ;       0100 = SERVICE MODE (SYSTEM TEST)
                            ;--------------------------------------------------------------
09BC                        E_MSG   PROC    NEAR
09BC  BA 0201                       MOV     DX,201H
09BF  EC                            IN      AL,DX           ; GET MODE BITS
09C0  24 F0                         AND     AL,0F0H         ; ISOLATE BITS OF INTEREST
09C2  75 03                         JNZ     EM0
09C4  E9 0A61 R                     JMP     MFG_OUT         ; MANUFACTURING MODE (BURN-IN)
09C7  3C 10                 EM0:    CMP     AL,00010000B    ;
09C9  75 03                         JNE     EM1
09CB  E9 0A61 R                     JMP     MFG_OUT         ; MFG. MODE (SYSTEM TEST)
09CE  8A F0                 EM1:    MOV     DH,AL           ; SAVE MODE
09D0  80 FF 0A                      CMP     BH,0AH          ; ERROR CODE ABOVE 0AH (CRT STARTED
                                                            ;  DISPLAY POSSIBLE)?
09D3  7C 63                         JL      BEEPS           ; DO BEEP OUTPUT IF BELOW 10H
09D5  53                            PUSH    BX              ; SAVE ERROR AND MODE FLAGS
09D6  56                            PUSH    SI
09D7  52                            PUSH    DX
09D8  B4 02                         MOV     AH,2            ; SET CURSOR
09DA  BA 1521                       MOV     DX,1521H        ; ROW 21, COL.33
09DD  B7 07                         MOV     BH,7            ; PAGE 7
09DF  CD 10                         INT     10H
09E1  BE 0030 R                     MOV     SI,OFFSET ERROR_ERR
09E4  B9 0005                       MOV     CX,5            ; PRINT WORD "ERROR"
09E7  2E: 8A 04             EM_0:   MOV     AL,CS:[SI]
09EA  46                            INC     SI
09EB  E8 18BA R                     CALL    PRT_HEX
09EE  E2 F7                         LOOP    EM_0
                            ; LOOK FOR A BLANK SPACE TO POSSIBLY PUT CUSTOMER LEVEL ERRORS (IN
                            ; CASE OF MULTI ERROR)
09F0  B6 16                         MOV     DH,16H
09F2  B4 02                 EM_1:   MOV     AH,2            ; SET CURSOR
09F4  CD 10                         INT     10H             ; ROW 22, COL33 (OR ABOVE, IF
                                                            ; MULTIPLE ERRS)
09F6  B4 08                         MOV     AH,8            ; READ CHARACTER THIS POSITION
09F8  CD 10                         INT     10H
09FA  FE C2                         INC     DL              ; POINT TO NEXT POSTION
09FC  3C 20                         CMP     AL,' '          ; BLANK?
09FE  75 F2                         JNE     EM_1            ; GO CHECK NEXT POSITION, IF NOT
0A00  5A                            POP     DX              ; RECOVER ERROR POINTERS
0A01  5E                            POP     SI
0A02  5B                            POP     BX
0A03  80 FE 20                      CMP     DH,00100000B    ; SERVICE MODE?
0A06  74 21                         JE      SERV_OUT        ;
0A08  80 FE 40                      CMP     DH,01000000B    ;
0A0B  74 1C                         JE      SERV_OUT
0A0D  2E: 8A 04                     MOV     AL,CS:[SI]      ; GET ERROR CHARACTER
0A10  E8 18BA R                     CALL    PRT_HEX         ; DISPLAY IT
0A13  80 FF 20                      CMP     BH,20H          ; ERROR BELOW 20? (MEM TROUBLE?)
0A16  7D 03                         JNL     EM_2
0A18  E9 0ABB R                     JMP     TOTLTPO         ; HALT SYSTEM IF SO.
                                    ASSUME  DS:XXDATA       ;
0A1B  1E                    EM_2:   PUSH    DS
0A1C  50                            PUSH    AX
0A1D  B8 ---- R                     MOV     AX,XXDATA
0A20  8E D8                         MOV     DS,AX
0A22  88 3E 0018 R                  MOV     POST_ERR,BH     ; SET ERROR FLAG NON-ZERO
0A26  58                            POP     AX
0A27  1F                            POP     DS
                                    ASSUME  DS:NOTHING
0A28  C3                            RET                     ; RETURN TO CALLER
```

```
0A29                          SERV_OUT:
0A29  8A C7                           MOV     AL,BH           ; PRINT MSB
0A2B  53                              PUSH    BX
0A2C  E8 18A9 R                       CALL    XPC_BYTE        ; DISPLAY IT
0A2F  5B                              POP     BX
0A30  8A C3                           MOV     AL,BL           ; PRINT LSB
0A32  E8 18A9 R                       CALL    XPC_BYTE
0A35  E9 0ABB R                       JMP     TOTLTPO
0A38  FA                      BEEPS:  CLI
0A39  8C C8                           MOV     AX,CS           ; SET CODE SEG= STACK SEG
0A3B  8E D0                           MOV     SS,AX           ; (STACK IS LOST, BUT THINGS ARE
                                                              ; OVER, ANYWAY)
0A3D  B2 02                           MOV     DL,2            ; 2 BEEPS
0A3F  BC 0028 R                       MOV     SP,OFFSET EX_0  ; SET DUMMY RETURN
0A42  B3 01                   EB:     MOV     BL,1            ; SHORT BEEP
0A44  E9 FF31 R                       JMP     BEEP            ;
0A47  E2 FE                   EB0:    LOOP    EB0             ; WAIT (BEEPER OFF)
0A49  FE CA                           DEC     DL              ; DONE YET?
0A4B  75 F5                           JNZ     EB              ; LOOP IF NOT
0A4D  80 FF 05                        CMP     BH,05H          ; 64K CARD ERROR?
0A50  75 69                           JNE     TOTLTPO         ; END IF NOT
0A52  80 FE 20                        CMP     DH,00100000B    ; SERVICE MODE?
0A55  74 05                           JE      EB1
0A57  80 FE 40                        CMP     DH,01000000B    ;
0A5A  75 5F                           JNE     TOTLTPO         ; END IF NOT
0A5C  B3 01                   EB1:    MOV     BL,1            ; ONE MORE BEEP FOR 64K ERROR IF IN
                                                              ; SERVICE MODE
0A5E  E9 FF31 R                       JMP     BEEP
0A61                          MFG_OUT:
0A61  FA                              CLI
0A62  E4 61                           IN      AL,PORT_B
0A64  24 FC                           AND     AL,0FCH
0A66  E6 61                           OUT     PORT_B,AL
0A68  BA 0011                         MOV     DX,11H          ; SEND DATA TO  ADDRESSES 11,12
0A6B  8A C7                           MOV     AL,BH           ;
0A6D  EE                              OUT     DX,AL           ; SEND HIGH BYTE
0A6E  42                              INC     DX
0A6F  8A C3                           MOV     AL,BL           ;
0A71  EE                              OUT     DX,AL           ; SEND LOW BYTE
                              ; INIT. ON-BOARD RS232 PORT FOR COMMUNICATIONS W/MFG MONITOR
                                      ASSUME  DS:XXDATA
0A72  B8 ---- R                       MOV     AX,XXDATA
0A75  8E D8                           MOV     DS,AX           ; POINT TO DATA SEGMENT CONTAINING
                                                              ; CHECKPOINT #
0A77  8C C8                           MOV     AX,CS
0A79  8E D0                           MOV     SS,AX           ; SET STACK FOR RTN
0A7B  BC 002E R                       MOV     SP,OFFSET EX1
0A7E  BA 02FB                         MOV     DX,02FBH        ; LINE CONTROL REG. ADDRESS
0A81  E9 F085 R                       JMP     S8250           ; GO SET UP FOR 9600, ODD, 2 STOP
                                                              ; BITS, 8 BITS
0A84  8B CA                   M01:    MOV     CX,DX           ; DX CAME BACK WITH XMIT REG
                                                              ; ADDRESS IN IT
0A86  BA 02FC                         MOV     DX,02FCH        ; MODEM CONTROL REG
0A89  2A C0                           SUB     AL,AL           ; SET DTR AND RTS LOW SO POSSIBLE
                                                              ; WRAP PLUG WON'T CONFUSE THINGS
0A8B  EE                              OUT     DX,AL
0A8C  BA 02FE                         MOV     DX,02FEH        ; MODEM STATUS REG
0A8F  EC                      M02:    IN      AL,DX
0A90  24 10                           AND     AL,00010000B    ; CTS UP YET?
0A92  74 FB                           JZ      M02             ; LOOP TILL IT IS
0A94  4A                              DEC     DX              ; SET DX=2FD (LINE STATUS REG)
0A95  87 D1                           XCHG    DX,CX           ; POINT TO XMIT. DATA REG
0A97  A0 0005 R                       MOV     AL,MFG_TST      ; GET MFG ROUTINE ERROR INDICATOR
0A9A  EE                              OUT     DX,AL           ; (MAY BE WRONG FOR EARLY ERRORS)
0A9B  EB 00                           JMP     $+2             ; DELAY
0A9D  87 D1                           XCHG    DX,CX           ; POINT DX=2FD
0A9F  EC                      M03:    IN      AL,DX           ; TRANSMIT EMPTY?
0AA0  24 20                           AND     AL,00100000B
0AA2  EB 00                           JMP     $+2             ; DELAY
0AA4  74 F9                           JZ      M03             ; LOOP TILL IT IS
0AA6  87 D1                           XCHG    DX,CX
0AA8  8A C7                           MOV     AL,BH           ; GET MSB OF ERROR WORD
0AAA  EE                              OUT     DX,AL
0AAB  EB 00                           JMP     $+2             ; DELAY
0AAD  87 D1                           XCHG    DX,CX
0AAF  EC                      M04:    IN      AL,DX           ; WAIT FOR XMIT EMPTY
0AB0  24 20                           AND     AL,00100000B
0AB2  EB 00                           JMP     $+2             ; DELAY
0AB4  74 F9                           JZ      M04
0AB6  8A C3                           MOV     AL,BL           ; GET LSB OF ERROR WORD
0AB8  87 D1                           XCHG    DX,CX
0ABA  EE                              OUT     DX,AL
0ABB                          TOTLTPO:
0ABB  FA                              CLI                     ; DISABLE INTS.
0ABC  2A C0                           SUB     AL,AL           ;
0ABE  E6 F2                           OUT     0F2H,AL         ; STOP DISKETTE MOTOR
0AC0  E6 A0                           OUT     0A0H,AL         ; DISABLE NMI
0AC2  F4                              HLT                     ; HALT
0AC3  C3                              RET
0AC4                          E_MSG   ENDP
```

```
                                ;-----------------------------------------------------------------
                                ;   SUBROUTINE TO INITIALIZE INS8250 PORTS TO THE MASTER RESET
                                ;       STATUS. THIS ROUTINE ALSO TESTS THE PORTS' PERMANENT
                                ;       ZERO BITS.
                                ;   EXPECTS TO BE PASSED:
                                ;       (DX) = ADDRESS OF THE 8250 TRANSMIT/RECEIVE BUFFER
                                ;   UPON RETURN:
                                ;       (CF) = 1  IF ONE OF THE PORTS' PERMANENT ZERO BITS WAS NOT
                                ;                 ZERO (ERR)
                                ;                 (DX) = PORT ADDRESS THAT FAILED TEST
                                ;                 (AL) = MEANINGLESS
                                ;                 (BL) = 2  INTR ENBL REG  BITS NOT 0
                                ;                        3  INTR ID REG BITS NOT 0
                                ;                        4  MODEM CTRL REG  BITS NOT 0
                                ;                        5  LINE STAT REG BITS NOT 0
                                ;              0  IF ALL PORTS' PERMANENT ZERO BITS WERE ZERO
                                ;                 (DX) = TRANSMIT/RECEIVE BUFFER ADDRESS
                                ;                 (AL) = LAST VALUE READ FROM RECEIVER BUFFER
                                ;                 (BL) = 5 (MEANINGLESS)
                                ;       PORTS SET UP AS FOLLOWS ON ERROR-FREE RETURN:
                                ;        XF9 - INTR ENBL REG  = 0          ALL INTERRUPTS DISABLED
                                ;        XFA - INTR ID REG    = 00000001B  NO INTERRUPTS PENDING
                                ;        XFB - LINE CTRL REG  = 0          ALL BITS LOW
                                ;        XFC - MODEM CTRL REG = 0          ALL BITS LOW
                                ;        XFD - LINE STAT REG  = 01100000B  TRANSMITTER HOLDING
                                ;                               REGISTER AND TRANSMITTER EMPTY ON
                                ;        XFE - MODEM STAT REG = XXXX0000B  WHERE X 'S REPRESENT
                                ;                               INPUT SIGNALS
                                ;   REGISTERS DX, AL, AND BL ARE ALTERED.  NO OTHER REGISTERS USED.
                                ;-----------------------------------------------------------------
0AC4                            I8250   PROC    NEAR
0AC4  EC                                IN      AL,DX           ; READ RECVR BUFFER BUT IGNORE
                                                                ; CONTENTS
0AC5  B3 02                             MOV     BL,2            ; ERROR INDICATOR
0AC7  E8 FE9F R                         CALL    RR2             ; READ INTR ENBL REG
0ACA  24 F0                             AND     AL,11110000B    ; BITS 4-7 OFF?
0ACC  75 28                             JNE     AT20            ; NO - ERROR
0ACE  E8 FE9A R                         CALL    RR1             ; READ INTR ID REG
0AD1  24 F8                             AND     AL,11111000B    ; BITS 3-7 OFF?
0AD3  75 21                             JNE     AT20            ; NO
0AD5  42                                INC     DX              ; LINE CTRL REG
0AD6  E8 FE9A R                         CALL    RR1             ; READ MODEM CTRL REG
0AD9  24 E0                             AND     AL,11100000B    ; BITS 5-7 OFF?
0ADB  75 19                             JNE     AT20            ; NO
0ADD  E8 FE9A R                         CALL    RR1             ; READ LINE STAT REG
0AE0  24 80                             AND     AL,10000000B    ; BIT 7 OFF?
0AE2  75 12                             JNE     AT20            ; NO
0AE4  B0 60                             MOV     AL,60H
0AE6  EE                                OUT     DX,AL
0AE7  EB 00                             JMP     $+2             ; I/O DELAY
0AE9  42                                INC     DX              ; MODEM STAT REG
0AEA  32 C0                             XOR     AL,AL
0AEC  EE                                OUT     DX,AL           ; WIRED BITS WILL BE HIGH
0AED  E8 FEA0 R                         CALL    RR3             ; CLEAR BITS 0-3 IN CASE THEY'RE ON
                                                                ; AFTER WRITING TO STATUS REG
0AF0  83 EA 06                          SUB     DX,6            ; RECEIVER BUFFER
0AF3  EC                                IN      AL,DX           ; IN CASE WRITING TO PORTS CAUSED
                                                                ; DATA READY TO GO HIGH!
0AF4  F8                                CLC
0AF5  C3                                RET
0AF6  F9                        AT20:   STC                     ; ERROR RETURN
0AF7  C3                                RET
0AF8                            I8250   ENDP
                                ;-----------------------------------------------------------------
                                ;   SUBROUTINE TO TEST A PARTICULAR 8250 INTERRUPT.  PASS IT THE
                                ;       (BIT # + 1) OF THE STATUS REGISTER THAT IS TO BE TESTED.
                                ;       THIS ROUTINE SETS THAT BIT AND CHECKS TO SEE IF THE CORRECT
                                ;       8250 INTERRUPT IS GENERATED.
                                ;   IT EXPECTS TO BE PASSED:
                                ;       (AH) = BIT # TO BE TESTED
                                ;       (BL) = INTERRUPT IDENTIFIER
                                ;           (0) = RECEIVED DATA AVAILABLE OR TRANSMITTER HOLDING
                                ;                 REGISTER EMPTY INTERRUPT TEST
                                ;           (1) = RECEIVER LINE STATUS OR MODEM STATUS INTERRUPT
                                ;                 TEST
                                ;       (BH) = BITS WHICH DETERMINE WHICH INTERRUPT IS TO BE
                                ;              CHECKED
                                ;           (0) = MODEM STATUS
                                ;           (2) = TRANSMITTER HOLDING REGISTER EMPTY
                                ;           (4) = RECEIVED DATA AVAILABLE
                                ;           (6) = RECEIVER LINE STATUS
                                ;       (CX) = VALUE TO SUBTRACT AND ADD IN ORDER TO REFERENCE THE
                                ;              INTERRUPT IDENTIFICATION REGISTER
                                ;           (3) = RECEIVED DATA AVAILABLE, TRANSMITTER HOLDING
                                ;                 REGISTER AND RECEIVER LINE STATUS INTERRUPTS
                                ;           (4) = MODEM STATUS INTERRUPT
                                ;       (DX) = ADDRESS OF THE LINE STATUS OR MODEM STATUS REGISTER
                                ;              IT RETURNS:
                                ;       (AL) = 0FFH IF TEST FAILS - EITHER NO INTERRUPT OCCURRED OR
                                ;              THE WRONG INTERRUPT OCCURRED
                                ;    OR
                                ;       (AL) = CONTENTS OF THE INTERRUPT ID REGISTER FOR RECEIVED
                                ;          DATA AVAILABLE AND TRANSMITTER HOLDING REGISTER
                                ;          EMPTY INTERRUPTS
                                ;          -OR-
                                ;          CONTENTS OF THE LINE STATUS OR MODEM STATUS REGISTER
                                ;              DEPENDING ON WHICH ONE WAS TESTED.
                                ;       (DX) = ADDRESS OF INTERRUPT ID REGISTER FOR RECEIVED DATA
                                ;              AVAILABLE OR TRANSMITTER HOLDING REGISTER EMPTY
                                ;              INTERRUPTS
                                ;    OR
                                ;       (DX) = ADDRESS OF THE LINE STATUS OR DATA SET STATUS
                                ;              REGISTER (DEPENDING ON WHICH INTERRUPT WAS TESTED)
                                ;   NO OTHER REGISTERS ARE ALTERED.
                                ;-----------------------------------------------------------------
```

```
0AF8                           ICT     PROC    NEAR
0AF8  EC                               IN      AL,DX           ; READ STATUS REGISTER
0AF9  EB 00                            JMP     $+2             ; I/O DELAY
0AFB  0A C4                            OR      AL,AH           ; SET TEST BIT
0AFD  EE                               OUT     DX,AL           ; WRITE IT TO THE STATUS REGISTER
0AFE  2B D1                            SUB     DX,CX           ; POINT TO INTERRUPT ID REGISTER
0B00  51                               PUSH    CX
0B01  2B C9                            SUB     CX,CX           ; WAIT FOR 8250 INTERRUPT TO OCCUR
0B03  EC                       AT21:   IN      AL,DX           ; READ INTR ID REG
0B04  A8 01                            TEST    AL,1            ; INTERRUPT PENDING?
0B06  74 02                            JE      AT22            ; YES -RETURN W/ INTERRUPT ID IN AL
0B08  E2 F9                            LOOP    AT21            ; NO - TRY AGAIN
0B0A  59                       AT22:   POP     CX              ; AL = 1 IF NO INTERRUPT OCCURRED
0B0B  3A C7                            CMP     AL,BH           ; INTERRUPT WE'RE LOOKING FOR?
0B0D  75 09                            JNE     AT23            ; NO
0B0F  0A DB                            OR      BL,BL           ; DONE WITH TEST FOR THIS INTERRUPT
0B11  74 07                            JE      AT24            ; RETURN W/ CONTENTS OF INTR ID REG
0B13  03 D1                            ADD     DX,CX           ; READ STATUS REGISTER TO CLEAR THE
0B15  EC                               IN      AL,DX           ; INTERRUPT (WHEN BL=1)
0B16  EB 02                            JMP     SHORT AT24      ; RETURN CONTENTS OF STATUS REG
0B18  B0 FF                    AT23:   MOV     AL,0FFH         ; SET ERROR INDICATOR
0B1A  C3                       AT24:   RET
0B1B                           ICT     ENDP
                               ;--- INT 19 ------------------------------------------
                               ; BOOT STRAP LOADER
                               ;       TRACK 0, SECTOR 1 IS READ INTO THE
                               ;       BOOT LOCATION (SEGMENT 0, OFFSET 7C00)
                               ;       AND CONTROL IS TRANSFERRED THERE.
                               ;
                               ;       IF THE DISKETTE IS NOT PRESENT OR HAS A
                               ;       PROBLEM LOADING (E.G., NOT READY), AN INT.
                               ;       18H IS EXECUITED. IF A CARTRIDGE HAS VECTORED
                               ;       INT. 18H TO ITSELF, CONTROL WILL BE PASSED TO
                               ;       THE CARTRIDGE.
                               ;-----------------------------------------------------
                                       ASSUME  CS:CODE,DS:ABS0
0B1B                           BOOT_STRAP      PROC    NEAR
0B1B  FB                               STI                     ; ENABLE INTERRUPTS
0B1C  2B C0                            SUB     AX,AX           ; SET 40X25 B&W MODE ON CRT
0B1E  CD 10                            INT     10H             ;
0B20  2B C0                            SUB     AX,AX           ; ESTABLISH ADDRESSING
0B22  8E D8                            MOV     DS,AX
                               ;----- SEE IF DISKETTE PRESENT
0B24  E4 62                            IN      AL,PORT_C       ; GET CONFIG BITS
0B26  24 04                            AND     AL,00000100B    ; IS DISKETTE PRESENT?
0B28  75 28                            JNZ     H3              ; NO, THEN ATTEMPT TO GO TO CART.
                               ;----- RESET THE DISK PARAMETER TABLE VECTOR
0B2A  C7 06 0078 R EFC7 R              MOV     WORD PTR DISK_POINTER, OFFSET DISK_BASE
0B30  8C 0E 007A R                     MOV     WORD PTR DISK_POINTER+2,CS
                               ;-----  LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT
0B34  B9 0004                          MOV     CX,4            ; SET RETRY COUNT
0B37  51                       H1:     PUSH    CX              ; SAVE RETRY COUNT
0B38  B4 00                            MOV     AH,0            ; RESET THE DISKETTE SYSTEM
0B3A  CD 13                            INT     13H             ; DISKETTE_IO
0B3C  72 0F                            JC      H2              ; IF ERROR, TRY AGAIN
0B3E  B8 0201                          MOV     AX,201H         ; READ IN THE SINGLE SECTOR
0B41  2B D2                            SUB     DX,DX           ; TO THE BOOT LOCATION
0B43  8E C2                            MOV     ES,DX
0B45  BB 7C00 R                        MOV     BX,OFFSET BOOT_LOCN
                                                               ; DRIVE 0, HEAD 0
0B48  B9 0001                          MOV     CX,1            ; SECTOR 1, TRACK 0
0B4B  CD 13                            INT     13H             ; DISKETTE_IO
0B4D  59                       H2:     POP     CX              ; RECOVER RETRY COUNT
0B4E  73 04                            JNC     H3A             ; CF SET BY UNSUCCESSFUL READ
0B50  E2 E5                            LOOP    H1              ; DO IT FOR RETRY TIMES
                               ;----- UNABLE TO IPL FROM THE DISKETTE
0B52  CD 18                    H3:     INT     18H             ; GO TO BASIC OR CARTRIDGE
                               ;----- IPL WAS SUCCESSFUL
0B54  EA 7C00 ---- R           H3A:    JMP     BOOT_LOCN
0B59                           BOOT_STRAP      ENDP
                               ;-----------------------------------------------------
                               ; THIS ROUTINE PERFORMS A READ/WRITE TEST ON A BLOCK OF
                               ; STORAGE (MAX. SIZE = 32KB). IF "WARM START", FILL
                               ; BLOCK WITH 0000 AND RETURN.
                               ; DATA PATTERNS USED:
                               ;       0->FF ON ONE BYTE TO TEST DATA BUS
                               ;       AAAA,5555,00FF,FF00 FOR ALL WORDS
                               ;       FILL WITH 0000 BEFORE EXIT
                               ; ON ENTRY:
                               ;   ES = ADDRESS OF STORAGE TO BE TESTED
                               ;   DS = ADDRESS OF STORAGE TO BE TESTED
                               ;   CX = WORD COUNT OF STORAGE BLOCK TO BE TESTED
                               ;        (MAX. = 8000H (32K WORDS))
                               ; ON EXIT:
                               ;   ZERO FLAG = OFF IF STORAGE ERROR
                               ;   IF ZERO FLAG = OFF, THEN CX = XOR'ED BIT PATTERN
                               ;   OF THE EXPECTED DATA PATTERN VS. THE ACTUAL DATA
                               ;   READ. (I.E., A BIT "ON" IN AL IS THE BIT IN ERROR)
                               ;   AH=03 IF BOTH BYTES OF WORD HAVE ERRORS
                               ;   AH=02 IF LOW (EVEN) BYTE HAS ERROR
                               ;   AH=01 IF HI (ODD) BYTE HAS ERROR
                               ; AX,BX,CX,DX,DI,SI ARE ALL DESTROYED.
                               ;-----------------------------------------------------
```

```
0B59                           PODSTG  PROC    NEAR
                                       ASSUME  DS:ABS0
0B59  FC                               CLD                     ; SET DIRECTION TO INCREMENT
0B5A  2B FF                            SUB     DI,DI           ; SET DI=0000 REL. TO START OF SEG
0B5C  2B C0                            SUB     AX,AX           ; INITIAL DATA PATTERN FOR 00-FF
                                                               ; TEST
0B5E  8E D8                            MOV     DS,AX           ; SET DS TO ABS0
0B60  8B 1E 0472 R                     MOV     BX,DATA_WORD[RESET_FLAG-DATA] ; WARM START?
0B64  81 FB 1234                       CMP     BX,1234H
0B68  8C C2                            MOV     DX,ES
0B6A  8E DA                            MOV     DS,DX           ; RESTORE DS
0B6C  75 0B                            JNE     P1
0B6E  F3/ AB                   P12:    REP     STOSW           ; SIMPLE FILL WITH 0 ON WARM-START
0B70  8E D8                            MOV     DS,AX
0B72  89 1E 0472 R                     MOV     DATA_WORD[RESET_FLAG-DATA],BX
0B76  8E DA                            MOV     DS,DX           ; RESTORE DS
0B78  C3                               RET                     ; AND EXIT
0B79  81 FB 4321               P1:     CMP     BX,4321H        ; DIAG. RESTART?
0B7D  74 EF                            JE      P12             ; DO FILL WITH ZEROS
0B7F  88 05                    P2:     MOV     [DI],AL         ; WRITE TEST DATA
0B81  8A 05                            MOV     AL,[DI]         ; GET IT BACK
0B83  32 C4                            XOR     AL,AH           ; COMPARE TO EXPECTED
0B85  74 03                            JZ      PY
0B87  E9 0C0C R                        JMP     P8              ; ERROR EXIT IF MISCOMPARE
0B8A  FE C4                    PY:     INC     AH              ; FORM NEW DATA PATTERN
0B8C  8A C4                            MOV     AL,AH           ;
0B8E  75 EF                            JNZ     P2              ; LOOP TILL ALL 256 DATA PATTERNS
                                                               ;  DONE
0B90  8B E9                            MOV     BP,CX           ; SAVE WORD COUNT
0B92  B8 AAAA                          MOV     AX,0AAAAH       ; LOAD DATA PATTERN
0B95  8B D8                            MOV     BX,AX
0B97  BA 5555                          MOV     DX,05555H       ; LOAD OTHER DATA PATTERN
0B9A  F3/ AB                           REP     STOSW           ; FILL WORDS FROM LOW TO HIGH
                                                               ; WITH AAAA
0B9C  4F                               DEC     DI              ; POINT TO LAST WORD WRITTEN
0B9D  4F                               DEC     DI
0B9E  FD                               STD                     ; SET DIRECTION FLAG TO GO DOWN
0B9F  8B F7                            MOV     SI,DI           ; SET INDEX REGS. EQUAL
0BA1  8B CD                            MOV     CX,BP           ; RECOVER WORD COUNT
0BA3                           P3:                             ; GO FROM HIGH TO LOW
0BA3  AD                               LODSW                   ; GET WORD FROM MEMORY
0BA4  33 C3                            XOR     AX,BX           ; EQUAL WHAT S/B THERE?
0BA6  75 64                            JNZ     P8              ; GO ERROR EXIT IF NOT
0BA8  8B C2                            MOV     AX,DX           ; GET 55 DATA PATTERN
0BAA  AB                               STOSW                   ;  STORE IT IN LOCATION JUST READ
0BAB  E2 F6                            LOOP    P3              ; LOOP TILL ALL BYTES DONE
0BAD  8B CD                            MOV     CX,BP           ; RECOVER WORD COUNT
0BAF  FC                               CLD                     ; BACK TO INCREMENT
0BB0  46                               INC     SI              ; ADJUST PTRS
0BB1  46                               INC     SI
0BB2  8B FE                            MOV     DI,SI
0BB4  8B DA                            MOV     BX,DX           ; S/B DATA PATTERN TO BX
0BB6  BA 00FF                          MOV     DX,00FFH        ; DATA FOR CHECKERBOARD PATTERN
0BB9  AD                       PX:     LODSW                   ; GET WORD FROM MEMORY
0BBA  33 C3                            XOR     AX,BX           ; EQUAL WHAT S/B THERE?
0BBC  75 4E                            JNZ     P8              ; GO ERROR EXIT IF NOT
0BBE  8B C2                            MOV     AX,DX           ; GET OTHER PATTERN
0BC0  AB                               STOSW                   ; STORE IT IN LOCATION JUST READ
0BC1  E2 F6                            LOOP    PX              ; LOOP TILL ALL BYTES DONE
0BC3  8B CD                            MOV     CX,BP           ; RECOVER WORD COUNT
0BC5  FD                               STD                     ; DECREMENT
0BC6  4E                               DEC     SI              ; ADJUST PTRS
0BC7  4E                               DEC     SI
0BC8  8B FE                            MOV     DI,SI
0BCA  8B DA                            MOV     BX,DX           ; S/B DATA PATTERN TO BX
0BCC  F7 D2                            NOT     DX              ; MAKE PATTERN FF00
0BCE  0A D2                            OR      DL,DL           ; FIRST PASS?
0BD0  74 E7                            JZ      PX
0BD2  FC                               CLD                     ; INCREMENT
0BD3  83 C6 04                         ADD     SI,4
0BD6  F7 D2                            NOT     DX
0BD8  8B FE                            MOV     DI,SI
0BDA  8B CD                            MOV     CX,BP
0BDC                           P4:                             ; LOW TO HIGH
0BDC  AD                               LODSW                   ; GET A WORD
0BDD  33 C2                            XOR     AX,DX           ; SHOULD COMPARE TO DX
0BDF  75 2B                            JNZ     P8              ; GO ERROR IF NOT
0BE1  AB                               STOSW                   ; WRITE 0000 BACK TO LOCATION
                                                               ; JUST READ
0BE2  E2 F8                            LOOP    P4              ; LOOP TILL DONE
0BE4  FD                               STD                     ; BACK TO DECREMENT
0BE5  4E                               DEC     SI              ; ADJUST POINTER DOWN TO LAST WORD
0BE6  4E                               DEC     SI              ; WRITTEN
                               ; CHECK IF IN SERVICE/MFG MODES, IF SO, PERFORM REFRESH CHECK
0BE7  BA 0201                          MOV     DX,201H         ;
0BEA  EC                               IN      AL,DX           ; GET OPTION BITS
0BEB  24 F0                            AND     AL,0F0H         ;
0BED  3C F0                            CMP     AL,0F0H         ; ALL BITS HIGH=NORMAL MODE
0BEF  74 10                            JE      P6
0BF1  8C C9                            MOV     CX,CS
0BF3  8C D3                            MOV     BX,SS
0BF5  3B CB                            CMP     CX,BX           ; SEE IF IN PRE-STACK MODE
0BF7  74 08                            JE      P6              ; BYPASS RETENTION TEST IF SO
0BF9  B0 18                            MOV     AL,24           ; SET OUTER LOOP COUNT
                               ; WAIT ABOUT 6-8 SECONDS WITHOUT ACCESSING MEMORY
                               ; IF REFRESH IS NOT WORKING PROPERLY, THIS SHOULD
                               ; BE ENOUGH TIME FOR SOME DATA TO GO SOUR.
```

```
0BFB  E2 FE                    P5:     LOOP    P5
0BFD  FE C8                            DEC     AL
0BFF  75 FA                            JNZ     P5
0C01  8B CD                    P6:     MOV     CX,BP           ; RECOVER WORD COUNT
0C03  AD                       P7:     LODSW                   ; GET WORD
0C04  0B C0                            OR      AX,AX           ; = TO 0000
0C06  75 04                            JNZ     P8              ; ERROR IF NOT
0C08  E2 F9                            LOOP    P7              ; LOOP TILL DONE
0C0A  EB 13                            JMP     SHORT P11       ; THEN EXIT
0C0C  8B C8                    P8:     MOV     CX,AX           ; SAVE BITS IN ERROR
0C0E  32 E4                            XOR     AH,AH
0C10  0A ED                            OR      CH,CH           ; HIGH BYTE ERROR?
0C12  74 02                            JZ      P9
0C14  FE C4                            INC     AH              ; SET HIGH BYTE ERROR
0C16  0A C9                    P9:     OR      CL,CL           ; LOW BYTE ERROR?
0C18  74 03                            JZ      P10
0C1A  80 C4 02                         ADD     AH,2
0C1D  0A E4                    P10:    OR      AH,AH           ; SET ZERO FLAG=0 (ERROR INDICATION
0C1F  FC                       P11:    CLD                     ; SET DIR FLAG BACK TO INCREMENT
0C20  C3                               RET                     ; RETURN TO CALLER
0C21                           PODSTG  ENDP
                               ;*****************************************************
                               ; PUT_LOGO PROCEDURE
                               ;       THIS PROC SETS UP POINTERS AND CALLS THE SCREEN
                               ;    OUTPUT ROUTINE SO THAT THE IBM LOGO, A MESSAGE,
                               ;    AND A COLOR BAR ARE PUT UP ON THE SCREEN.
                               ;    AX,BX, AND DX ARE DESTROYED. ALL OTHERS ARE SAVED
                               ;*****************************************************
0C21                           PUT_LOGO PROC   NEAR
0C21  1E                               PUSH    DS
0C22  55                               PUSH    BP
0C23  50                               PUSH    AX
0C24  53                               PUSH    BX
0C25  51                               PUSH    CX
0C26  52                               PUSH    DX
0C27  BD 0C4A R                        MOV     BP,OFFSET LOGO  ;
0C2A  BA 8000                          MOV     DX,8000H       ;POINT DH DL AT ROW,COLUMN 0,0
0C2D  B3 1F                            MOV     BL,00011111B   ;ATTRIBUTE OF CHARACTERS TO BE
                                                              ;WRITTEN
0C2F  CD 82                            INT     82H            ;CALL OUTPUT ROUTINE
0C31  B3 00                            MOV     BL,00000000B    ; INITIALIZE ATTRIBUTE
0C33  B2 00                            MOV     DL,0            ; INITIALIZE COLUMN
0C35  B6 94                    AGAIN:  MOV     DH,94H          ; SET LINE
0C37  BD 0CDD R                        MOV     BP,OFFSET COLOR ; OUTPUT GIVEN COLOR BAR
0C3A  CD 82                            INT     82H            ;CALL OUTPUT ROUTINE
0C3C  FE C3                            INC     BL              ; INCREMENT ATTRIBUTE
0C3E  80 FA 20                         CMP     DL,32           ; IS THE COLUMN COUNTER POINTING
                                                               ; PAST 40?
0C41  7C F2                            JL      AGAIN           ; IF NOT, DO IT AGAIN
0C43  5A                               POP     DX
0C44  59                               POP     CX
0C45  5B                               POP     BX
0C46  58                               POP     AX
0C47  5D                               POP     BP              ; RESTORE BP
0C48  1F                               POP     DS              ; RESTORE DS
0C49  C3                               RET
0C4A                           PUT_LOGO ENDP
0C4A  03                       LOGO    DB      LOGO_E - LOGO
0C4B  20 DC                            DB      ' ',220
= 0C4D                         LOGO_E  =       $
0C4D  28 FB                            DB      40,-5
0C4F  28 FB                            DB      40,-5
0C51  02 07 01 09 03 04                DB      2,7,1,9,3,4,9,4,1,-5
      09 04 01 FB
0C5B  02 07 01 0A 02 05                DB      2,7,1,10,2,5,7,5,1,-5
      07 05 01 FB
0C65  02 07 01 0B 01 06                DB      2,7,1,11,1,6,5,6,1,-5
      05 06 01 FB
0C6F  04 03 05 03 03 03                DB      4,3,5,3,3,3,3,5,3,5,3,-5
      03 05 03 05 03 FB
0C7B  04 03 05 03 03 03                DB      4,3,5,3,3,3,3,6,1,6,3,-5
      03 06 01 06 03 FB
0C87  04 03 05 08 04 0D                DB      4,3,5,8,4,13,3,-5
      03 FB
0C8F  04 03 05 07 05 0D                DB      4,3,5,7,5,13,3,-5
      03 FB
0C97  04 03 05 08 04 0D                DB      4,3,5,8,4,13,3,-5
      03 FB
0C9F  04 03 05 03 03 03                DB      4,3,5,3,3,3,3,13,3,-5
      03 0D 03 FB
0CA9  04 03 05 03 03 03                DB      4,3,5,3,3,3,3,3,1,5,1,3,3,-5
      03 03 01 05 01 03
      03 FB
0CB7  02 07 01 0B 01 05                DB      2,7,1,11,1,5,2,3,2,5,1,-5
      02 03 02 05 01 FB
0CC3  02 07 01 0A 02 05                DB      2,7,1,10,2,5,3,1,3,5,1,-5
      03 01 03 05 01 FB
0CCF  02 07 01 09 03 05                DB      2,7,1,9,3,5,7,5,1,-5
      07 05 01 FB
0CD9  28 FB                            DB      40,-5
0CDB  28 FC                            DB      40,-4
0CDD  02                       COLOR   DB      COLOR_E - COLOR
0CDE  DB                               DB      219
= 0CDF                         COLOR_E =       $
0CDF  02 77 02 77 02 77                DB      2,121-2,2,121-2,2,121-2,2,121-2,2,-4
      02 77 02 FC
                                       ASSUME  DS:DATA
```

```
;--- INT 10 -----------------------------------------------------
; VIDEO_IO
;      THESE ROUTINES PROVIDE THE CRT INTERFACE
;      THE FOLLOWING FUNCTIONS ARE PROVIDED:
;      (AH)=0  SET MODE (AL) CONTAINS MODE VALUE
;              (AL)=0 40X25 BW (POWER ON DEFAULT)
;              (AL)=1 40X25 COLOR
;              (AL)=2  80X25 BW
;              (AL)=3  80X25 COLOR
;              GRAPHICS MODES
;              (AL)=4  320X200 4 COLOR
;              (AL)=5  320X200 BW 4 SHADES
;              (AL)=6  640X200 BW 2 SHADES
;              (AL)=7  NOT VALID
;       **** EXTENDED MODES ***
;              (AL)=8  160X200 16 COLOR
;              (AL)=9  320X200 16 COLOR
;              (AL)=A  640X200 4 COLOR
;              *** NOTE BW MODES OPERATE SAME AS COLOR MODES, BUT
;                      COLOR BURST IS NOT ENABLED
;              *** NOTE IF HIGH ORDER BIT IN AL IS SET, THE REGEN
;                      BUFFER IS NOT CLEARED.
;      (AH)=1  SET CURSOR TYPE
;              (CH) =  BITS 4-0 = START LINE FOR CURSOR
;              ** HARDWARE WILL ALWAYS CAUSE BLINK
;              ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC
;                 BLINKING OR NO CURSOR AT ALL
;              ** IN GRAPHICS MODES, BIT 5 IS FORCED ON TO
;                 DISABLE THE CURSOR
;              (CL) =  BITS 4-0 = END LINE FOR CURSOR
;      (AH)=2  SET CURSOR POSITION
;              (DH,DL) = ROW,COLUMN  (0,0) IS UPPER LEFT
;              (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
;      (AH)=3  READ CURSOR POSITION
;              (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
;              ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR
;                      (CH,CL) = CURSOR MODE CURRENTLY SET
;      (AH)=4  READ LIGHT PEN POSITION
;              ON EXIT:
;           (AH) = 0 -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED
;           (AH) = 1 -- VALID LIGHT PEN VALUE IN REGISTERS
;                      (DH,DL) = ROW,COLUMN OF CHARACTER LP POSN
;                      (CH) = RASTER LINE (0-199)
;                      (BX) = PIXEL COLUMN (0-319,639)
;      (AH)=5  SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR
;                     ALPHA MODES)
;              (AL)=NEW PAGE VALUE (0-7 FOR MODES 0&1, 0-3 FOR
;                     MODES 2&3)
;              IF BIT 7 (80H) OF AL=1
;                      READ/WRITE CRT/CPU PAGE REGISTERS
;              (AL) = 80H READ CRT/CPU PAGE REGISTERS
;              (AL) = 81H SET CPU PAGE REGISTER
;                      (BL) = VALUE TO SET
;              (AL) = 82H SET CRT PAGE REGISTER
;                      (BH) = VALUE TO SET
;              (AL) = 83H SET BOTH CRT AND CPU PAGE REGISTERS
;                      (BL) = VALUE TO SET IN CPU PAGE REGISTER
;                      (BH) = VALUE TO SET IN CRT PAGE REGISTER
;              IF BIT 7 (80H) OF AL=1
;                 ALWAYS  RETURNS (BH) = CONTENTS OF CRT PAGE REG
;                                 (BL) = CONTENTS OF CPU PAGE REG
;      (AH)=6  SCROLL ACTIVE PAGE UP
;              (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT
;                     BOTTOM OF WINDOW, AL = 0 MEANS BLANK
;                     ENTIRE WINDOW
;              (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF
;                        SCROLL
;              (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF
;                        SCROLL
;              (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
;      (AH)=7  SCROLL ACTIVE PAGE DOWN
;              (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP
;                     OF WINDOW, AL=0 MEANS BLANK ENTIRE WINDOW
;              (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF
;                        SCROLL
;              (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF
;                        SCROLL
;              (BH) = ATTRIBUTE TO BE USED ON BLANK LINE
;
;      CHARACTER HANDLING ROUTINES
;      (AH) = 8 READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION
;              (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
;              ON EXIT:
;              (AL) = CHAR READ
;              (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES
;                     ONLY)
;      (AH) = 9 WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR
;             POSITION
;             (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
;             (CX) = COUNT OF CHARACTERS TO WRITE
;             (AL) = CHAR TO WRITE
;             (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF
;                     CHARACTER (GRAPHICS). SEE NOTE ON WRITE
;                     DOT FOR BIT 7 OF BL = 1.
;      (AH) = 10 (0AH) WRITE CHARACTER ONLY AT CURRENT CURSOR
;                      POSITION
;              (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY)
;              (CX) = COUNT OF CHARACTERS TO WRITE
;              (AL) = CHAR TO WRITE
;              (BL) = COLOR OF CHAR (GRAPHICS)
;                      SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.
;       FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE,
;               THE CHARACTERS ARE FORMED FROM A CHARACTER
;               GENERATOR IMAGE MAINTAINED IN THE SYSTEM ROM.
;               INTERRUPT 44H (LOCATION 00110H) IS USED TO
;               POINT TO THE 1K BYTE TABLE CONTAINING THE
;               FIRST 128 CHARS (0-127).
;               INTERRUPT 1FH (LOCATION 0007CH) IS USED TO
;               POINT TO THE 1K BYTE TABLE CONTAINING THE SECOND
;               128 CHARS (128-255).
;       FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE
;               REPLICATION FACTOR CONTAINED IN (CX) ON ENTRY WILL
;               PRODUCE VALID RESULTS ONLY FOR CHARACTERS
;               CONTAINED ON THE SAME ROW.  CONTINUATION TO
;               SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY.
;
;       GRAPHICS INTERFACE
;       (AH) = 11 (0BH) SET COLOR PALETTE
;               (BH) = PALETTE COLOR ID BEING SET (0-127)
;               (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID
;                       COLOR ID = 0 SELECTS THE BACKGROUND
;                                    COLOR (0-15)
;                       COLOR ID = 1 SELECTS THE PALETTE TO BE
;                          USED:
;                          2 COLOR MODE:
;                               0 = WHITE FOR COLOR 1
;                               1 = BLACK FOR COLOR 1
;                          4 COLOR MODES:
;                               0 = GREEN, RED, BROWN FOR
;                                   COLORS 1,2,3
;                               1 = CYAN, MAGENTA, WHITE FOR
;                                   COLORS 1,2,3
;                          16 COLOR MODES:
;                               ALWAYS SETS UP PALETTE AS:
;                               BLUE FOR COLOR 1
;                               GREEN FOR COLOR 2
;                               CYAN FOR COLOR 3
;                               RED FOR COLOR 4
;                               MAGENTA FOR COLOR 5
;                               BROWN FOR COLOR 6
;                               LIGHT GRAY FOR COLOR 7
;                               DARK GRAY FOR COLOR 8
;                               LIGHT BLUE FOR COLOR 9
;                               LIGHT GREEN FOR COLOR 10
;                               LIGHT CYAN FOR COLOR 11
;                               LIGHT RED FOR COLOR 12
;                               LIGHT MAGENTA FOR COLOR 13
;                               YELLOW FOR COLOR 14
;                               WHITE FOR COLOR 15
;                IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET
;                       FOR PALETTE COLOR 0 INDICATES THE BORDER
;                       COLOR TO BE USED. IN GRAPHIC MODES, IT
;                       INDICATES THE BORDER COLOR AND THE
;                       BACKGROUND COLOR.
;       (AH) = 12 (0CH) WRITE DOT
;               (DX) = ROW NUMBER
;               (CX) = COLUMN NUMBER
;               (AL) = COLOR VALUE
;                  IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS
;                  EXCLUSIVE OR'D WITH THE CURRENT CONTENTS OF
;                  THE DOT
;       (AH) = 13 (0DH) READ DOT
;               (DX) = ROW NUMBER
;               (CX) = COLUMN NUMBER
;               (AL) RETURNS THE DOT READ
;
; ASCII TELETYPE ROUTINE FOR OUTPUT
;       (AH) = 14 (0EH) WRITE TELETYPE TO ACTIVE PAGE
;               (AL) = CHAR TO WRITE
;               (BL) = FOREGROUND COLOR IN GRAPHICS MODE
;               NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS
;                       MODE SET
;       (AH) = 15 (0FH) CURRENT VIDEO STATE
;               RETURNS THE CURRENT VIDEO STATE
;               (AL) = MODE CURRENTLY SET (SEE AH=0 FOR
;                      EXPLANATION)
;               (AH) = NUMBER OF CHARACTER COLUMNS ON SCREEN
;               (BH) = CURRENT ACTIVE DISPLAY PAGE
;       (AH) = 16 (10H) SET PALETTE REGISTERS
;               (AL) = 0 SET PALETTE REGISTER
;                       (BL) = PALETTE REGISTER TO SET (00H - 0FH)
;                       (BH) = VALUE TO SET
;               (AL) = 1 SET BORDER COLOR REGISTER
;                       (BH) = VALUE TO SET
;               (AL) = 2 SET ALL PALETTE REGISTERS AND BORDER
;                        REGISTER
;                       ES:DX POINTS TO A 17 BYTE LIST
;                       BYTES 0 THRU 15 ARE VALUES FOR PALETTE
;                               REGISTERS 0 THRU 15
;                       BYTE 16 IS THE VALUE FOR THE BORDER
;                               REGISTER
;       NOTE:
;       IN MODES USING A 32K REGEN (9 AND A), ACCESS THROUGH THE CPU
;       REGISTER BY USE OF B800H SEGMENT VALUE ONLY REACHES THE
;       FIRST 16K. BIOS USES THE CONTENTS OF THE CPU PAGE REG
;       (BITS 3,4, & 5 OF PAGDAT IN BIOS DATA AREA) TO DERIVE THE
;       PROPER SEGMENT VALUE.
;
;       CS,SS,DS,ES,BX,CX,DX PRESERVED DURING CALL
;       ALL OTHERS DESTROYED
```

```
                               ;-----------------------------------------------------
                               ; VIDEO GATE ARRAY REGISTERS
                               ;
                               ;  PORT 3DA OUTPUT
                               ;       REG 0   MODE CONTROL 1 REGISTER
                               ;       01H     +HI BANDWIDTH/-LOW BANDWIDTH
                               ;       02H     +GRAPHICS/-ALPHA
                               ;       04H     +B&W
                               ;       08H     +VIDEO ENABLE
                               ;       10H     +16 COLOR GRAPHICS
                               ;
                               ;       REG 1   PALETTE MASK REISTER
                               ;       01H     PALETTE MASK 0
                               ;       02H     PALETTE MASK 1
                               ;       04H     PALETTE MASK 2
                               ;       08H     PALETTE MASK 3
                               ;
                               ;       REG 2   BORDER COLOR REGISTER
                               ;       01H     BLUE
                               ;       02H     GREEN
                               ;       04H     RED
                               ;       08H     INTENSITY
                               ;
                               ;       REG 3   MODE CONTROL 2 REGISTER
                               ;       01H     RESERVED -- MUST BE ZERO
                               ;       02H     +ENABLE BLINK
                               ;       04H     RESERVED -- MUST BE ZERO
                               ;       08H     +2 COLOR GRAPHICS (640X200 2 COLOR ONLY)
                               ;
                               ;       REG 4   RESET REGISTER
                               ;       01H     +ASYNCHRONOUS RESET
                               ;       02H     +SYNCHRONOUS RESET
                               ;
                               ;       REGS 10 TO 1F   PALETTE REGISTERS
                               ;       01H     BLUE
                               ;       02H     GREEN
                               ;       04H     RED
                               ;       08H     INTENSITY
                               ;
                               ;  VIDEO GATE ARRAY STATUS
                               ;    PORT 3DA INPUT
                               ;       01H     +DISPLAY ENABLE
                               ;       02H     +LIGHT PEN TRIGGER SET
                               ;       04H     -LIGHT PEN SWITCH MADE
                               ;       08H     +VERTICAL RETRACE
                               ;       10H     +VIDEO DOTS
                                       ASSUME  CS:CODE,DS:DATA,ES:VIDEO_RAM
0CE9                           M0010   LABEL   WORD    ;  TABLE OF ROUTINES WITHIN VIDEO I/O
0CE9  0DA5 R                           DW      OFFSET  SET_MODE
0CEB  E45E R                           DW      OFFSET  SET_CTYPE
0CED  E488 R                           DW      OFFSET  SET_CPOS
0CEF  E52D R                           DW      OFFSET  READ_CURSOR
0CF1  F751 R                           DW      OFFSET  READ_LPEN
0CF3  E4B3 R                           DW      OFFSET  ACT_DISP_PAGE
0CF5  E5D3 R                           DW      OFFSET  SCROLL_UP
0CF7  E63F R                           DW      OFFSET  SCROLL_DOWN
0CF9  F0E4 R                           DW      OFFSET  READ_AC_CURRENT
0CFB  F113 R                           DW      OFFSET  WRITE_AC_CURRENT
0CFD  F12C R                           DW      OFFSET  WRITE_C_CURRENT
0CFF  E543 R                           DW      OFFSET  SET_COLOR
0D01  F187 R                           DW      OFFSET  WRITE_DOT
0D03  F146 R                           DW      OFFSET  READ_DOT
0D05  1992 R                           DW      OFFSET  WRITE_TTY
0D07  E5B1 R                           DW      OFFSET  VIDEO_STATE
0D09  E685 R                           DW      OFFSET  SET_PALLETTE
= 0022                         M0010L  EQU     $-M0010

0D0B                           VIDEO_IO        PROC    NEAR
0D0B  FB                               STI                     ; INTERRUPTS BACK ON
0D0C  FC                               CLD                     ; SET DIRECTION FORWARD
0D0D  06                               PUSH    ES
0D0E  1E                               PUSH    DS              ; SAVE SEGMENT REGISTERS
0D0F  52                               PUSH    DX
0D10  51                               PUSH    CX
0D11  53                               PUSH    BX
0D12  56                               PUSH    SI
0D13  57                               PUSH    DI
0D14  50                               PUSH    AX              ; SAVE AX VALUE
0D15  8A C4                            MOV     AL,AH           ; GET INTO LOW BYTE
0D17  32 E4                            XOR     AH,AH           ; ZERO TO HIGH BYTE
0D19  D1 E0                            SAL     AX,1            ; *2 FOR TABLE LOOKUP
0D1B  8B F0                            MOV     SI,AX           ; PUT INTO SI FOR BRANCH
0D1D  3D 0022                          CMP     AX,M0010L       ; TEST FOR WITHIN RANGE
0D20  72 04                            JB      C1              ; BRANCH AROUND BRANCH
0D22  58                               POP     AX              ; THROW AWAY THE PARAMETER
0D23  E9 0F70 R                        JMP     VIDEO_RETURN    ; DO NOTHING IF NOT IN RANGE
0D26  E8 138B R                C1:     CALL    DDS
0D29  B8 B800                          MOV     AX,0B800H       ; SEGMENT FOR COLOR CARD
0D2C  80 3E 0049 R 09                  CMP     CRT_MODE,9      ; IN MODE USING 32K REGEN
0D31  72 09                            JC      C2              ; NO,JUMP
0D33  8A 26 008A R                     MOV     AH,PAGDAT       ; GET COPY OF PAGE REGS
0D37  80 E4 38                         AND     AH,CPUREG       ; ISOLATE CPU REG
0D3A  D0 EC                            SHR     AH,1            ; SHIFT TO MAKE INTO SEGMENT VALUE
0D3C  8E C0                    C2:     MOV     ES,AX           ; SET UP TO POINT AT VIDEO RAM AREA
0D3E  58                               POP     AX              ; RECOVER VALUE
0D3F  8A 26 0049 R                     MOV     AH,CRT_MODE     ; GET CURRENT MODE INTO AH
0D43  2E: FF A4 0CE9 R                 JMP     WORD PTR CS:[SI+OFFSET M0010]
0D48                           VIDEO_IO        ENDP
```

Appendix A

```
                               ;----------------------------------------
                               ; SET_MODE
                               ;        THIS ROUTINE INITIALIZES THE ATTACHMENT TO
                               ;        THE SELECTED MODE.  THE SCREEN IS BLANKED.
                               ; INPUT
                               ;        (AL) = MODE SELECTED (RANGE 0-B)
                               ; OUTPUT
                               ;        NONE
                               ;-----------------------------------------
0D48                           M0050   LABEL   WORD            ; TABLE OF REGEN LENGTHS
0D48  0800                             DW      2048            ; MODE 0 40X25  BW
0D4A  0800                             DW      2048            ; MODE 1 40X25  COLOR
0D4C  1000                             DW      4096            ; MODE 2 80X25  BW
0D4E  1000                             DW      4096            ; MODE 3 80X25  COLOR
0D50  4000                             DW      16384           ; MODE 4 320X200 4 COLOR
0D52  4000                             DW      16384           ; MODE 5 320X200 4 COLOR
0D54  4000                             DW      16384           ; MODE 6 640X200 BW
0D56  0000                             DW      0               ; MODE 7 INVALID
0D58  4000                             DW      16384           ; MODE 8 160X200 16 COLOR
0D5A  8000                             DW      32768           ; MODE 9 320X200 16 COLOR
0D5C  8000                             DW      32768           ; MODE A 640X200 4 COLOR
                               ;------ COLUMNS
0D5E                           M0060   LABEL   BYTE
0D5E  28 28 50 50 28 28                DB      40,40,80,80,40,40,80,0,20,40,80
      50 00 14 28 50
                               ;------- TABLE OF GATE ARRAY PARAMATERS FOR MODE SETTING
0D69                           M0070   LABEL   BYTE
                               ;------ SET UP FOR 40X25 BW     MODE 0
0D69  0C 0F 00 02                      DB      0CH,0FH,0,2     ; GATE ARRAY PARMS
= 0004                         M0070L  EQU     $-M0070
                               ;------ SET UP FOR 40X25 COLOR  MODE 1
0D6D  08 0F 00 02                      DB      08H,0FH,0,2     ; GATE ARRAY PARMS
                               ;------ SET UP FOR 80X25 BW     MODE 2
0D71  0D 0F 00 02                      DB      0DH,0FH,0,2     ; GATE ARRAY PARMS
                               ;------ SET UP FOR 80X25 COLOR  MODE 3
0D75  09 0F 00 02                      DB      09H,0FH,0,2     ; GATE ARRAY PARMS
                               ;------ SET UP FOR 320X200  4 COLOR     MODE 4
0D79  0A 03 00 00                      DB      0AH,03H,0,0     ; GATE ARRAY PARMS
                               ;------ SET UP FOR 320X200  BW          MODE 5
0D7D  0E 03 00 00                      DB      0EH,03H,0,0     ; GATE ARRAY PARMS
                               ;------ SET UP FOR 640X200  BW          MODE 6
0D81  0E 01 00 08                      DB      0EH,01H,0,8     ; GATE ARRAY PARMS
                               ;------ SET UP FOR    INVALID           MODE 7
0D85  00 00 00 00                      DB      00H,00H,0,0     ; GATE ARRAY PARMS
                               ;------ SET UP FOR 160X200  16 COLOR    MODE 8
0D89  1A 0F 00 00                      DB      1AH,0FH,0,0     ; GATE ARRAY PARMS
                               ;------ SET UP FOR 320X200  16 COLOR    MODE 9
0D8D  1B 0F 00 00                      DB      1BH,0FH,0,0     ; GATE ARRAY PARMS
                               ;------ SET UP FOR 640X200   4 COLOR    MODE A
0D91  0B 03 00 00                      DB      0BH,03H,0,0     ; GATE ARRAY PARMS
                               ;-------------- TABLES OF PALETTE COLORS FOR 2 AND 4 COLOR MODES
                               ;-----   2 COLOR, SET 0
0D95                           M0072   LABEL   BYTE
0D95  00 0F 00 00                      DB      0,0FH,0,0
= 0004                         M0072L  EQU     $-M0072         ;ENTRY LENGTH
                               ;-----   2 COLOR, SET 1
0D99  0F 00 00 00                      DB      0FH,0,0,0
                               ;-----   4 COLOR, SET 0
0D9D                           M0074   LABEL   BYTE
0D9D  00 02 04 06                      DB      0,2,4,6
                               ;-----   4 COLOR, SET 1
0DA1                           M0075   LABEL   BYTE
0DA1  00 03 05 0F                      DB      0,3,5,0FH
0DA5                           SET_MODE        PROC    NEAR
0DA5  50                               PUSH    AX              ;SAVE INPUT MODE ON STACK
0DA6  24 7F                            AND     AL,7FH          ;REMOVE CLEAR REGEN SWITCH
0DA8  3C 07                            CMP     AL,7            ;CHECK FOR VALID MODES
0DAA  74 04                            JE      C3              ;MODE 7 IS INVALID
0DAC  3C 0B                            CMP     AL,0BH
0DAE  72 02                            JC      C4              ;GREATER THAN A IS INVALID
0DB0  B0 00                    C3:     MOV     AL,0            ;DEFAULT TO MODE 0
0DB2  3C 02                    C4:     CMP     AL,2            ;CHECK FOR MODES NEEDING 128K
0DB4  74 08                            JE      C5
0DB6  3C 03                            CMP     AL,3
0DB8  74 04                            JE      C5
0DBA  3C 09                            CMP     AL,09H
0DBC  72 0A                            JC      C6
0DBE  81 3E 0015 R 0080        C5:     CMP     TRUE_MEM,128    ;DO WE HAVE 128K?
0DC4  73 02                            JNC     C6              ;YES, JUMP
0DC6  B0 00                            MOV     AL,0            ;NO, DEFAULT TO MODE 0
0DC8  BA 03D4                  C6:     MOV     DX,03D4H        ; ADDRESS OF COLOR CARD
0DCB  8A E0                            MOV     AH,AL           ; SAVE MODE IN AH
0DCD  A2 0049 R                        MOV     CRT_MODE,AL     ; SAVE IN GLOBAL VARIABLE
0DD0  89 16 0063 R                     MOV     ADDR_6845,DX    ; SAVE ADDRESS OF BASE
0DD4  8B F8                            MOV     DI,AX           ; SAVE MODE IN DI
0DD6  BA 03DA                          MOV     DX,VGA_CTL      ; POINT TO CONTROL REGISTER
0DD9  EC                               IN      AL,DX           ; SYNC CONTROL REG TO ADDRESS
0DDA  32 C0                            XOR     AL,AL           ; SET VGA REG 0
0DDC  EE                               OUT     DX,AL           ; SELECT IT
0DDD  A0 0065 R                        MOV     AL,CRT_MODE_SET ; GET LAST MODE SET
0DE0  24 F7                            AND     AL,0F7H         ; TURN OFF VIDEO
0DE2  EE                               OUT     DX,AL           ; SET IN GATE ARRAY
```

```
                                         ;----- SET DEFAULT PALETTES
0DE3  8B C7                                      MOV     AX,DI            ; GET MODE
0DE5  B4 10                                      MOV     AH,10H           ; SET PALETTE REG 0
0DE7  BB 0D95 R                                  MOV     BX,OFFSET M0072  ; POINT TO TABLE ENTRY
0DEA  3C 06                                      CMP     AL,6             ; 2 COLOR MODE?
0DEC  74 0F                                      JE      C7               ; YES, JUMP
0DEE  BB 0DA1 R                                  MOV     BX,OFFSET M0075  ; POINT TO TABLE ENTRY
0DF1  3C 05                                      CMP     AL,5             ; CHECK FOR 4 COLOR MODE
0DF3  74 08                                      JE      C7               ; YES, JUMP
0DF5  3C 04                                      CMP     AL,4             ; CHECK FOR 4 COLOR MODE
0DF7  74 04                                      JE      C7               ; YES JUMP
0DF9  3C 0A                                      CMP     AL,0AH           ; CHECK FOR 4 COLOR MODE
0DFB  75 11                                      JNE     C9               ; NO, JUMP
0DFD  B9 0004                            C7:     MOV     CX,4             ; NUMBER OF REGS TO SET
0E00  8A C4                              C8:     MOV     AL,AH            ; GET REG NUMBER
0E02  EE                                         OUT     DX,AL            ; SELECT IT
0E03  2E: 8A 07                                  MOV     AL,CS:[BX]       ; GET DATA
0E06  EE                                         OUT     DX,AL            ; SET IT
0E07  FE C4                                      INC     AH               ; NEXT REG
0E09  43                                         INC     BX               ; NEXT TABLE VALUE
0E0A  E2 F4                                      LOOP    C8
0E0C  EB 0B                                      JMP     SHORT C11
                                         ;----- SET PALETTES FOR DEFAULT 16 COLOR
0E0E  B9 0010                            C9:     MOV     CX,16            ; NUMBER OF PALETTES, AH IS REG
                                                                          ; COUNTER
0E11  8A C4                              C10:    MOV     AL,AH            ; GET REG NUMBER
0E13  EE                                         OUT     DX,AL            ; SELECT IT
0E14  EE                                         OUT     DX,AL            ; SET PALETTE VALUE
0E15  FE C4                                      INC     AH               ; NEXT REG
0E17  E2 F8                                      LOOP    C10
                                         ;----- SET UP M0 & M1 IN PAGREG
0E19  8B C7                              C11:    MOV     AX,DI            ; GET CURRENT MODE
0E1B  32 DB                                      XOR     BL,BL            ; SET UP FOR ALPHA MODE
0E1D  3C 04                                      CMP     AL,4             ; IN ALPHA MODE
0E1F  72 08                                      JC      C12              ; YES, JUMP
0E21  B3 40                                      MOV     BL,40H           ; SET UP FOR 16K REGEN
0E23  3C 09                                      CMP     AL,09H           ; MODE USE 16K
0E25  72 02                                      JC      C12              ; YES, JUMP
0E27  B3 C0                                      MOV     BL,0C0H          ; SET UP FOR 32K REGEN
0E29  BA 03DF                            C12:    MOV     DX,PAGREG        ; SET PORT ADDRESS OF PAGREG
0E2C  A0 008A R                                  MOV     AL,PAGDAT        ; GET LAST DATA OUTPUT
0E2F  24 3F                                      AND     AL,3FH           ; CLEAR M0 & M1 BITS
0E31  0A C3                                      OR      AL,BL            ; SET NEW BITS
0E33  EE                                         OUT     DX,AL            ; STUFF BACK IN PORT
0E34  A2 008A R                                  MOV     PAGDAT,AL        ; SAVE COPY IN RAM
                                         ;----- ENABLE VIDEO AND CORRECT PORT SETTING
0E37  8B C7                                      MOV     AX,DI            ; GET CURRENT MODE
0E39  32 E4                                      XOR     AH,AH            ;   INTO AX REG
0E3B  B9 0004                                    MOV     CX,M0070L        ; SET TABLE ENTRY LENGTH
0E3E  F7 E1                                      MUL     CX               ; TIMES MODE FOR OFFSET INTO TABLE
0E40  8B D8                                      MOV     BX,AX            ; TABLE OFFSET IN BX
0E42  81 C3 0D69 R                               ADD     BX,OFFSET M0070  ; ADD TABLE START TO OFFSET
0E46  2E: 8A 27                                  MOV     AH,CS:[BX]       ; SAVE MODE SET AND PALETTE
0E49  2E: 8A 47 02                               MOV     AL,CS:[BX + 2]   ;   TILL WE CAN PUT THEM IN RAM
0E4D  8B F0                                      MOV     SI,AX
0E4F  FA                                         CLI                      ; DISABLE INTERRUPTS
0E50  E8 E675 R                                  CALL    MODE_ALIVE       ; KEEP MEMORY DATA VALID
0E53  B0 10                                      MOV     AL,10H           ; DISABLE NMI AND HOLD REQUEST
0E55  E6 A0                                      OUT     NMI_PORT,AL      ;
0E57  BA 03DA                                    MOV     DX,VGA_CTL       ;
0E5A  B0 04                                      MOV     AL,4             ; POINT TO RESET REG
0E5C  EE                                         OUT     DX,AL            ; SEND TO GATE ARRAY
0E5D  B0 02                                      MOV     AL,2             ; SET SYNCHRONOUS RESET
0E5F  EE                                         OUT     DX,AL            ; DO IT
                                         ; WHILE THE GATE ARRAY IS IN RESET STATE, WE CANNOT ACCESS RAM
0E60  8B C6                                      MOV     AX,SI            ; RESTORE NEW MODE SET
0E62  80 E4 F7                                   AND     AH,0F7H          ; TURN OFF VIDEO ENABLE
0E65  32 C0                                      XOR     AL,AL            ; SET UP TO SELECT VGA REG 0
0E67  EE                                         OUT     DX,AL            ; SELECT IT
0E68  86 E0                                      XCHG    AH,AL            ; AH IS VGA REG COUNTER
0E6A  EE                                         OUT     DX,AL            ; SET MODE
0E6B  B0 04                                      MOV     AL,4             ; SET UP TO SELECT VGA REG 4
0E6D  EE                                         OUT     DX,AL            ; SELECT IT
0E6E  32 C0                                      XOR     AL,AL            ;
0E70  EE                                         OUT     DX,AL            ; REMOVE RESET FROM VGA
                                         ; NOW OKAY TO ACCESS RAM AGAIN
0E71  B0 80                                      MOV     AL,80H           ; ENABLE NMI AGAIN
0E73  E6 A0                                      OUT     NMI_PORT,AL      ;
0E75  E8 E675 R                                  CALL    MODE_ALIVE       ; KEEP MEMORY DATA VALID
0E78  FB                                         STI                      ; ENABLE INTERRUPTS
0E79  EB 07                                      JMP     SHORT C14
0E7B  8A C4                              C13:    MOV     AL,AH            ; GET VGA REG NUMBER
0E7D  EE                                         OUT     DX,AL            ; SELECT REG
0E7E  2E: 8A 07                                  MOV     AL,CS:[BX]       ; GET TABLE VALUE
0E81  EE                                         OUT     DX,AL            ; PUT IN VGA REG
0E82  43                                 C14:    INC     BX               ; NEXT IN TABLE
0E83  FE C4                                      INC     AH               ; NEXT REG
0E85  E2 F4                                      LOOP    C13              ; DO ENTIRE ENTRY
                                         ;---- SET UP CRT AND CPU PAGE REGS ACCORDING TO MODE & MEMORY SIZE
0E87  BA 03DF                                    MOV     DX,PAGREG        ; SET IO ADDRESS OF PAGREG
0E8A  A0 008A R                                  MOV     AL,PAGDAT        ; GET LAST DATA OUTPUT
0E8D  24 C0                                      AND     AL,0C0H          ; CLEAR REG BITS
0E8F  B3 36                                      MOV     BL,36H           ; SET UP FOR GRAPHICS MODE WITH 32K
                                                                          ; REGEN
0E91  A8 80                                      TEST    AL,80H           ; IN THIS MODE?
0E93  75 0C                                      JNZ     C15              ; YES, JUMP
0E95  B3 3F                                      MOV     BL,3FH           ; SET UP FOR 16K REGEN AND 128K
                                                                          ; MEMORY
0E97  81 3E 0015 R 0080                          CMP     TRUE_MEM,128     ; DO WE HAVE 128K?
0E9D  73 02                                      JNC     C15              ; YES, JUMP
0E9F  B3 1B                                      MOV     BL,1BH           ; SET UP FOR 16K REGEN AND 64K
                                                                          ; MEMORY
```

```
0EA1  0A C3                 C15:    OR      AL,BL              ; COMBINE MODE BITS AND REG VALUES
0EA3  EE                            OUT     DX,AL              ; SET PORT
0EA4  A2 008A R                     MOV     PAGDAT,AL          ; SAVE COPY IN RAM
0EA7  8B C6                         MOV     AX,SI              ; PUT MODE SET & PALETTE IN RAM
0EA9  88 26 0065 R                  MOV     CRT_MODE_SET,AH
0EAD  A2 0066 R                     MOV     CRT_PALLETTE,AL
0EB0  E4 61                         IN      AL,PORT_B          ; GET CURRENT VALUE OF 8255 PORT B
0EB2  24 FB                         AND     AL,0FBH            ; SET UP GRAPHICS MODE
0EB4  F6 C4 02                      TEST    AH,2               ; JUST SET ALPHA MODE IN VGA?
0EB7  75 02                         JNZ     C16                ; YES, JUMP
0EB9  0C 04                         OR      AL,4               ; SET UP ALPHA MODE
0EBB  E6 61                 C16:    OUT     PORT_B,AL          ; STUFF BACK IN 8255
                            ;------- SET UP 6845
0EBD  1E                            PUSH    DS                 ; SAVE DATA SEGMENT VALUE
0EBE  33 C0                         XOR     AX,AX              ; SET UP FOR ABSO SEGMENT
0EC0  8E D8                         MOV     DS,AX              ; ESTABLISH VECTOR TABLE ADDRESSING
                                    ASSUME  DS:ABSO
0EC2  C5 1E 0074 R                  LDS     BX,PARM_PTR        ; GET POINTER TO VIDEO PARMS
                                    ASSUME  DS:CODE
0EC6  8B C7                         MOV     AX,DI              ; GET CURRENT MODE IN AX
0EC8  B9 0010 90                    MOV     CX,M0040           ; LENGTH OF EACH ROW OF TABLE
0ECC  80 FC 02                      CMP     AH,2               ; DETERMINE WHICH TO USE
0ECF  72 10                         JC      C17                ; MODE IS 0 OR 1
0ED1  03 D9                         ADD     BX,CX              ; MOVE TO NEXT ROW OF INIT TABLE
0ED3  80 FC 04                      CMP     AH,4
0ED6  72 09                         JC      C17                ; MODE IS 2 OR 3
0ED8  03 D9                         ADD     BX,CX              ; MOVE TO GRAPHICS ROW OF
                                                               ; INIT_TABLE
0EDA  80 FC 09                      CMP     AH,9
0EDD  72 02                         JC      C17                ; MODE IS 4, 5, 6, 8, OR 9
0EDF  03 D9                         ADD     BX,CX              ; MOVE TO NEXT GRAPHICS ROW OF
                                                               ; INIT_TABLE
                            ;----- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
0EE1  50                    C17:    PUSH    AX                 ; SAVE MODE IN AH
0EE2  8A 47 02                      MOV     AL,DS:[BX+2]       ; GET HORZ. SYNC POSITION
0EE5  8B 7F 0A                      MOV     DI,WORD PTR DS:[BX+10]  ; GET CURSOR TYPE
0EE8  1E                            PUSH    DS
0EE9  E8 138B R                     CALL    DDS
                                    ASSUME  DS:DATA
0EEC  A2 0089 R                     MOV     HORZ_POS,AL        ; SAVE HORZ. SYNC POSITION VARIABLE
0EEF  89 3E 0060 R                  MOV     CURSOR_MODE,DI     ; SAVE CURSOR MODE
0EF3  50                            PUSH    AX
0EF4  A0 0086 R                     MOV     AL,VAR_DELAY       ; SET DEFAULT OFFSET
0EF7  24 0F                         AND     AL,0FH
0EF9  A2 0086 R                     MOV     VAR_DELAY,AL
0EFC  58                            POP     AX
                                    ASSUME  DS:CODE
0EFD  1F                            POP     DS
0EFE  32 E4                         XOR     AH,AH              ; AH WILL SERVE AS REGISTER NUMBER
                                                               ; DURING LOOP
0F00  BA 03D4                       MOV     DX,03D4H           ; POINT TO 6845
                            ;LOOP THROUGH TABLE, OUTPUTTING REG ADDRESS, THEN VALUE FROM TABLE
0F03  8A C4                 C18:    MOV     AL,AH              ; GET 6845 REGISTER NUMBER
0F05  EE                            OUT     DX,AL
0F06  42                            INC     DX                 ; POINT TO DATA PORT
0F07  FE C4                         INC     AH                 ; NEXT REGISTER VALUE
0F09  8A 07                         MOV     AL,[BX]            ; GET TABLE VALUE
0F0B  EE                            OUT     DX,AL              ; OUT TO CHIP
0F0C  43                            INC     BX                 ; NEXT IN TABLE
0F0D  4A                            DEC     DX                 ; BACK TO POINTER REGISTER
0F0E  E2 F3                         LOOP    C18                ; DO THE WHOLE TABLE
0F10  58                            POP     AX                 ; GET MODE BACK
0F11  1F                            POP     DS                 ; RECOVER SEGMENT VALUE
                                    ASSUME  DS:DATA
                            ;------ FILL REGEN AREA WITH BLANK
0F12  33 FF                         XOR     DI,DI              ; SET UP POINTER FOR REGEN
0F14  89 3E 004E R                  MOV     CRT_START,DI       ; START ADDRESS SAVED IN GLOBAL
0F18  C6 06 0062 R 00               MOV     ACTIVE_PAGE,0      ; SET PAGE VALUE
0F1D  5A                            POP     DX                 ; GET ORIGINAL INPUT BACK
0F1E  80 E2 80                      AND     DL,80H             ; NO CLEAR OF REGEN ?
0F21  75 1C                         JNZ     C21                ; SKIP CLEARING REGEN
0F23  BA B800                       MOV     DX,0B800H          ; SET UP SEGMENT FOR 16K REGEN AREA
0F26  B9 2000                       MOV     CX,8192            ; NUMBER OF WORDS TO CLEAR
0F29  3C 09                         CMP     AL,09H             ; REQUIRE 32K BYTE REGEN ?
0F2B  72 05                         JC      C19                ; NO, JUMP
0F2D  D1 E1                         SHL     CX,1               ; SET 16K WORDS TO CLEAR
0F2F  BA 1800                       MOV     DX,1800H           ; SET UP SEGMENT FOR 32K REGEN AREA
0F32  8E C2                 C19:    MOV     ES,DX              ; SET REGEN SEGMENT
0F34  3C 04                         CMP     AL,4               ; TEST FOR GRAPHICS
0F36  B8 0F20                       MOV     AX,' '+15*256      ; FILL CHAR FOR ALPHA
0F39  72 02                         JC      C20                ; NO_GRAPHICS_INIT
0F3B  33 C0                         XOR     AX,AX              ; FILL FOR GRAPHICS MODE
0F3D  F3/ AB                C20:    REP     STOSW              ; FILL THE REGEN BUFFER WITH BLANKS
                            ;----- ENABLE VIDEO
0F3F  BA 03DA               C21:    MOV     DX,VGA_CTL         ; SET PORT ADDRESS OF VGA
0F42  32 C0                         XOR     AL,AL              ;
0F44  EE                            OUT     DX,AL              ; SELECT VGA REG 0
0F45  A0 0065 R                     MOV     AL,CRT_MODE_SET    ; GET MODE SET VALUE
0F48  EE                            OUT     DX,AL              ; SET MODE
                            ;----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
                            ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE
0F49  32 FF                         XOR     BH,BH
0F4B  8A 1E 0049 R                  MOV     BL,CRT_MODE
0F4F  2E: 8A 87 0D5E R              MOV     AL,CS:[BX + OFFSET M0060]
0F54  32 E4                         XOR     AH,AH
0F56  A3 004A R                     MOV     CRT_COLS,AX        ; NUMBER OF COLUMNS IN THIS SCREEN
```

```
                               ;----- SET CURSOR POSITIONS
0F59  D1 E3                            SHL     BX,1            ; WORD OFFSET INTO CLEAR LENGTH
                                                               ; TABLE
0F5B  2E: 8B 8F 0D48 R                 MOV     CX,CS:[BX + OFFSET M0050]  ; LENGTH TO CLEAR
0F60  89 0E 004C R                     MOV     CRT_LEN,CX      ; SAVE LENGTH OF CRT
0F64  B9 0008                          MOV     CX,8            ; CLEAR ALL CURSOR POSITIONS
0F67  BF 0050 R                        MOV     DI,OFFSET CURSOR_POSN
0F6A  1E                               PUSH    DS              ; ESTABLISH SEGMENT
0F6B  07                               POP     ES              ;   ADDRESSING
0F6C  33 C0                            XOR     AX,AX
0F6E  F3/ AB                           REP     STOSW           ; FILL WITH ZEROES
                               ;----- NORMAL RETURN FROM ALL VIDEO RETURNS
0F70                           VIDEO_RETURN:
0F70  5F                               POP     DI
0F71  5E                               POP     SI
0F72  5B                               POP     BX
0F73  59                       C22:    POP     CX
0F74  5A                               POP     DX
0F75  1F                               POP     DS
0F76  07                               POP     ES              ; RECOVER SEGMENTS
0F77  CF                               IRET                    ; ALL DONE
0F78                           SET_MODE        ENDP
                               ;-----------------------------------------------------
                               ;
                               ;
                               ; KBDNMI - KEYBOARD NMI INTERRUPT ROUTINE
                               ;
                               ;     THIS ROUTINE OBTAINS CONTROL UPON AN NMI INTERRUPT, WHICH
                               ;     OCCURS UPON A KEYSTROKE FROM THE KEYBOARD.
                               ;
                               ;     THIS ROUTINE WILL DE-SERIALIZE THE BIT STREAM IN ORDER TO
                               ;     GET THE KEYBOARD SCAN CODE ENTERED.  IT THEN ISSUES INT 41
                               ;     PASSING THE SCAN CODE IN AL TO THE KEY PROCESSOR. UPON RETURN
                               ;     IT RE-ENABLES NMI AND RETURNS TO SYSTEM (IRET).
                               ;
                               ;-----------------------------------------------------
                                           ASSUME CS:CODE,DS:DATA
0F78                           KBDNMI  PROC    FAR
                               ;-----------DISABLE INTERRUPTS
0F78  FA                               CLI
                               ;-----------SAVE REGS & DISABLE NMI
0F79  56                               PUSH    SI
0F7A  57                               PUSH    DI
0F7B  50                               PUSH    AX              ; SAVE REGS
0F7C  53                               PUSH    BX
0F7D  51                               PUSH    CX
0F7E  52                               PUSH    DX
0F7F  1E                               PUSH    DS
0F80  06                               PUSH    ES
                               ;-----------INIT COUNTERS
0F81  BE 0008                          MOV     SI,8            ; SET UP # OF DATA BITS
0F84  32 DB                            XOR     BL,BL           ; INIT. PARITY COUNTER
                               ;-----------SAMPLE 5 TIMES TO  VALIDATE START BIT
0F86  32 E4                            XOR     AH,AH
0F88  B9 0005                          MOV     CX,5            ; SET COUNTER
0F8B  E4 62                    I1:     IN      AL,PORT_C       ; GET SAMPLE
0F8D  A8 40                            TEST    AL,40H          ; TEST IF 1
0F8F  74 02                            JZ      I2              ; JMP IF 0
0F91  FE C4                            INC     AH              ; KEEP COUNT OF 1'S
0F93  E2 F6                    I2:     LOOP    I1              ; KEEP SAMPLING
0F95  80 FC 03                         CMP     AH,3            ; VALID START BIT ?
0F98  73 03                            JNB     I25             ; JUMP IF OK
0F9A  EB 5D 90                         JMP     I8              ; INVALID (SYNC ERROR) NO AUDIO
                                                               ; OUTPUT
                               ;-----------VALID START BIT, LOOK FOR TRAILING EDGE
0F9D  B9 0032                  I25:    MOV     CX,50           ; SET UP WATCHDOG TIMEOUT
0FA0  E4 62                    I3:     IN      AL,PORT_C       ; GET SAMPLE
0FA2  A8 40                            TEST    AL,40H          ; TEST IF 0
0FA4  74 05                            JZ      I5              ; JMP IF TRAILING EDGE FOUND
0FA6  E2 F8                            LOOP    I3              ; KEEP LOOKING FOR TRAILING EDGE
0FA8  EB 4F 90                         JMP     I8              ; SYNC ERROR (STUCK ON 1'S)
                               ;-----------READ CLOCK TO SET START OF BIT TIME
0FAB  B0 40                    I5:     MOV     AL,40H          ; READ CLOCK
0FAD  E6 43                            OUT     TIM_CTL,AL      ; *
0FAF  90                               NOP                     ; *
0FB0  90                               NOP                     ; *
0FB1  E4 41                            IN      AL,TIMER+1      ; *
0FB3  8A E0                            MOV     AH,AL           ; *
0FB5  E4 41                            IN      AL,TIMER+1      ; *
0FB7  86 E0                            XCHG    AH,AL           ; *
0FB9  8B F8                            MOV     DI,AX           ; SAVE CLOCK TIME IN DI
                               ;-----------VERIFY VALID TRANSITION
0FBB  B9 0004                          MOV     CX,4            ; SET COUNTER
0FBE  E4 62                    I6:     IN      AL,PORT_C       ; GET SAMPLE
0FC0  A8 40                            TEST    AL,40H          ; TEST IF 0
0FC2  75 35                            JNZ     I8              ; JMP IF INVALID TRANSITION (SYNC)
0FC4  E2 F8                            LOOP    I6              ; KEEP LOOKING FOR VALID TRANSITION
                               ;-----------SET UP DISTANCE TO MIDDLE OF 1ST DATA BIT
0FC6  BA 0220                          MOV     DX,544          ; 310 USEC AWAY (.838 US / CT)
                               ;-------START LOOKING FOR TIME TO READ DATA BITS AND ASSEMBLE BYTE
0FC9  E8 1031 R                I7:     CALL    I30
0FCC  BA 020E                          MOV     DX,526          ; SET NEW DISTANCE TO NEXT HALF BIT
0FCF  50                               PUSH    AX              ; SAVE 1ST HALF BIT
0FD0  E8 1031 R                        CALL    I30
0FD3  8A C8                            MOV     CL,AL           ; PUT 2ND HALF BIT IN CL
0FD5  58                               POP     AX              ; RESTORE 1ST HALF BIT
0FD6  3A C8                            CMP     CL,AL           ; ARE THEY OPPOSITES ?
0FD8  74 2A                            JE      I9              ; NO, PHASE ERROR
```

```
                                ;-----------VALID DATA BIT, PLACE IN SCAN BYTE
0FDA  D0 EF                             SHR     BH,1            ; SHIFT PREVIOUS BITS
0FDC  0A F8                             OR      BH,AL           ; OR IN NEW DATA BIT
0FDE  4E                                DEC     SI              ; DECREMENT DATA BIT COUNTER
0FDF  75 E8                             JNZ     I7              ; CONTINUE FOR MORE DATA BITS
                                ;-----------WAIT FOR TIME TO SAMPLE PARITY BIT
0FE1  E8 1031 R                         CALL    I30
0FE4  50                                PUSH    AX              ; SAVE 1ST HALF BIT
0FE5  E8 1031 R                         CALL    I30
0FE8  8A C8                             MOV     CL,AL           ; PUT 2ND HALF BIT IN CL
0FEA  58                                POP     AX              ; RESTORE 1ST HALF BIT
0FEB  3A C8                             CMP     CL,AL           ; ARE THEY OPPOSITES ?
0FED  74 15                             JE      I9              ; NO, PHASE ERROR
                                ;-----------VALID PARITY BIT, CHECK PARITY
0FEF  80 E3 01                          AND     BL,1            ; CHECK IF ODD PARITY
0FF2  74 10                             JZ      I9              ; JMP IF PARITY ERROR
                                ;-----------VALID CHARACTER, SEND TO CHARACTER PROCESSING
0FF4  FB                                STI                     ; ENABLE INTERRUPTS
0FF5  8A C7                             MOV     AL,BH           ; PLACE SCAN CODE IN AL
0FF7  CD 48                             INT     48H             ; CHARACTER PROCESSING
                                ;-----------RESTORE REGS AND RE-ENABEL NMI
0FF9  07                        I8:     POP     ES              ; RESTORE REGS
0FFA  1F                                POP     DS
0FFB  5A                                POP     DX
0FFC  59                                POP     CX
0FFD  5B                                POP     BX
0FFE  E4 A0                             IN      AL,0A0H         ; ENABLE NMI
1000  58                                POP     AX
1001  5F                                POP     DI
1002  5E                                POP     SI
1003  CF                                IRET                    ; RETURN TO SYSTEM
                                ;-----------PARITY, SYNCH OR PHASE ERROR. OUTPUT MISSED KEY BEEP
1004  E8 138B R                 I9:     CALL    DDS             ; SETUP ADDRESSING
1007  83 FE 08                          CMP     SI,8            ; ARE WE ON THE FIRST DATA BIT?
100A  74 ED                             JE      I8              ; NO AUDIO FEEDBACK (MIGHT BE A
                                                                ; ..GLITCH)
100C  F6 06 0018 R 01                   TEST    KB_FLAG_1,01H   ; CHECK IF TRANSMISSION ERRORS
                                                                ; ..ARE TO BE REPORTED
1011  75 18                             JNZ     I10             ; 1=DO NOT BEEP, 0=BEEP
1013  BB 0080                           MOV     BX,080H         ; DURATION OF ERROR BEEP
1016  B9 0048                           MOV     CX,048H         ; FREQUENCY OF ERROR BEEP
1019  E8 E035 R                         CALL    KB_NOISE        ; AUDIO FEEDBACK
101C  80 26 0017 R F0                   AND     KB_FLAG,0F0H    ; CLEAR ALT,CLRL,LEFT AND RIGHT
                                                                ; SHIFTS
1021  80 26 0018 R 0F                   AND     KB_FLAG_1,0FH   ; CLEAR POTENTIAL BREAK OF INS,CAPS
                                                                ; NUM AND SCROLL SHIFT
1026  80 26 0088 R 1F                   AND     KB_FLAG_2,1FH   ; CLEAR FUNCTION STATES
102B  FE 06 0012 R              I10:    INC     KBD_ERR         ; KEEP TRACK OF KEYBOARD ERRORS
102F  EB C8                             JMP     SHORT I8        ; RETURN FROM INTERRUPT
1031                            KBDNMI  ENDP
1031                            I30     PROC    NEAR
1031  B0 40                     I31:    MOV     AL,40H          ; READ CLOCK
1033  E6 43                             OUT     TIM_CTL,AL      ; *
1035  90                                NOP                     ; *
1036  90                                NOP                     ; *
1037  E4 41                             IN      AL,TIMER+1      ; *
1039  8A E0                             MOV     AH,AL           ; *
103B  E4 41                             IN      AL,TIMER+1      ; *
103D  86 E0                             XCHG    AH,AL           ; *
103F  8B CF                             MOV     CX,DI           ; GET LAST CLOCK TIME
1041  2B C8                             SUB     CX,AX           ; SUB CURRENT TIME
1043  3B CA                             CMP     CX,DX           ; IS IT TIME TO SAMPLE ?
1045  72 EA                             JC      I31             ; NO, KEEP LOOKING AT TIME
1047  2B CA                             SUB     CX,DX           ; UPDATE # OF COUNTS OFF
1049  8B F8                             MOV     DI,AX           ; SAVE CURRENT TIME AS LAST TIME
104B  03 F9                             ADD     DI,CX           ; ADD DIFFERENCE FOR NEXT TIME
                                ;-----------START SAMPLING DATA BIT (5 SAMPLES)
104D  B9 0005                           MOV     CX,5            ; SET COUNTER
                                ;----------------------------------------------------------
                                ;
                                ; SAMPLE LINE
                                ;
                                ;       PORT_C IS SAMPLED CX TIMES AND IF THER ARE 3 OR MORE 1"S
                                ;       THEN 80H IS RETURNED IN AL, ELSE 00H IS RETURNED IN AL.
                                ;       PARITY COUNTER IS MAINTAINED IN ES.
                                ;
                                ;-----------------------------------------------------------
1050  32 E4                             XOR     AH,AH           ; CLEAR COUNTER
1052  E4 62                     I32:    IN      AL,PORT_C       ; GET SAMPLE
1054  A8 40                             TEST    AL,40H          ; TEST IF 1
1056  74 02                             JZ      I33             ; JMP IF 0
1058  FE C4                             INC     AH              ; KEEP COUNT OF 1'S
105A  E2 F6                     I33:    LOOP    I32             ; KEEP SAMPLING
105C  80 FC 03                          CMP     AH,3            ; VALID 1 ?
105F  72 05                             JB      I34             ; JMP IF NOT VALID 1
1061  B0 80                             MOV     AL,080H         ; RETURN 80H IN AL (1)
1063  FE C3                             INC     BL              ; INCREMENT PARITY COUNTER
1065  C3                                RET                     ; RETURN TO CALLER
1066  32 C0                     I34:    XOR     AL,AL           ; RETURN 0 IN AL (0)
1068  C3                                RET                     ; RETURN TO CALLER
1069                            I30     ENDP
```

```
                                        ;-------------------------------------------------------------
                                        ;KEY62_INT
                                        ;       THE PURPOSE OF THIS ROUTINE IS TO TRANSLATE SCAN CODES AND
                                        ;       SCAN CODE COMBINATIONS FROM THE 62 KEY KEYBOARD TO THEIR
                                        ;       EQUIVILENTS ON THE 83 KEY KEYBOARD.  THE SCAN CODE IS
                                        ;       PASSED IN AL. EACH SCAN CODE PASSED EITHER TRIGGERS ONE OR
                                        ;       MORE CALLS TO INTERRUPT 9 OR SETS FLAGS TO RETAIN KEYBOARD
                                        ;       STATUS.  WHEN INTERRUPT 9 IS CALLED THE TRANSLATED SCAN
                                        ;       CODES ARE PASSED TO IT IN AL.  THE INTENT OF THIS CODE WAS
                                        ;       TO KEEP INTERRUPT 9 INTACT FROM ITS ORIGIN IN THE PC FAMILY
                                        ;       THIS ROUTINE IS IN THE FRONT END OF INTERRUPT 9 AND
                                        ;       TRANSFORMS A 62 KEY KEYBOARD TO LOOK AS IF IT WERE AN 83
                                        ;       KEY VERSION.
                                        ;       IT IS ASSUMED THAT THIS ROUTINE IS CALLED FROM THE NMI
                                        ;       DESERIALIZATION ROUTINE AND THAT ALL REGISTERS WERE SAVED
                                        ;       IN THE CALLING ROUTINE.  AS A CONSEQUENCE ALL REGISTERS ARE
                                        ;       DESTROYED.
                                        ;-------------------------------------------------------------
                                        ;EQUATES
 = 0080                                 BREAK_BIT       EQU     80H
 = 0054                                 FN_KEY          EQU     54H
 = 0055                                 PHK             EQU     FN_KEY+1
 = 0056                                 EXT_SCAN        EQU     PHK+1    ; BASE CODE FOR SCAN CODES
                                                                         ; EXTENDING BEYOND 83
 = 00FF                                 AND_MASK        EQU     0FFH     ; USED TO SELECTIVELY REMOVE BITS
 = 001F                                 CLEAR_FLAGS     EQU     AND_MASK - (FN_FLAG+FN_BREAK+FN_PENDING)
                                        ;SCAN CODES
 = 0030                                 B_KEY           EQU     48
 = 0010                                 Q_KEY           EQU     16
 = 0019                                 P_KEY           EQU     25
 = 0012                                 E_KEY           EQU     18
 = 001F                                 S_KEY           EQU     31
 = 0031                                 N_KEY           EQU     49
 = 0048                                 UP_ARROW        EQU     72
 = 0050                                 DOWN_ARROW      EQU     80
 = 004B                                 LEFT_ARROW      EQU     75
 = 004D                                 RIGHT_ARROW     EQU     77
 = 000C                                 MINUS           EQU     12
 = 000D                                 EQUALS          EQU     13
 = 000B                                 NUM_0           EQU     11
                                        ;NEW TRANSLATED SCAN CODES
                                        ;----------------------------------------------------
                                        ;NOTE:
                                        ;       BREAK, PAUSE, ECHO, AND PRT_SCREEN ARE USED AS OFFSETS
                                        ;       INTO THE TABLE 'SCAN'. OFFSET = TABLE POSITION + 1.
                                        ;----------------------------------------------------
 = 0001                                 ECHO            EQU     01
 = 0002                                 BREAK           EQU     02
 = 0003                                 PAUSE           EQU     03
 = 0004                                 PRT_SCREEN      EQU     04
 = 0046                                 SCROLL_LOCK     EQU     70
 = 0045                                 NUM_LOCK        EQU     69
 = 0047                                 HOME            EQU     71
 = 004F                                 END_KEY         EQU     79
 = 0049                                 PAGE_UP         EQU     73
 = 0051                                 PAGE_DOWN       EQU     81
 = 004A                                 KEYPAD_MINUS    EQU     74
 = 004E                                 KEYPAD_PLUS     EQU     78
                                                ASSUME CS:CODE,DS:DATA
                                        ;----TABLE OF VALID SCAN CODES
 1069                                   KB0     LABEL BYTE
 1069  30 10 12 19 1F 31                        DB B_KEY, Q_KEY, E_KEY, P_KEY, S_KEY, N_KEY
 106F  48 50 4B 4D 0C                           DB UP_ARROW, DOWN_ARROW, LEFT_ARROW, RIGHT_ARROW, MINUS
 1074  0D                                       DB EQUALS
 = 000C                                 KB0LEN          EQU $ - KB0
                                        ;----TABLE OF NEW SCAN CODES
 1075                                   KB1     LABEL BYTE
 1075  02 03 01 04 46 45                        DB BREAK, PAUSE, ECHO, PRT_SCREEN, SCROLL_LOCK, NUM_LOCK
 107B  47 4F 49 51 4A 4E                        DB HOME,END_KEY,PAGE_UP,PAGE_DOWN,KEYPAD_MINUS,KEYPAD_PLUS
                                        ;-----------------------------------------
                                        ;NOTE:  THERE IS A ONE TO ONE CORRESPONDENCE BETWEEN
                                        ;       THE SIZE OF KB0 AND KB1.
                                        ;-----------------------------------------
                                        ;TABLE OF NUMERIC KEYPAD SCAN CODES
                                        ;       THESE SCAN CODES WERE NUMERIC KEYPAD CODES ON
                                        ;       THE 83 KEY KEYBOARD.
                                        ;----------------------------------------------
 1081                                   NUM_CODES LABEL BYTE
 1081  4F 50 51 4B 4C 4D                        DB 79,80,81,75,76,77,71,72,73,82
       47 48 49 52
                                        ;----------------------------------------------
                                        ;TABLE OF SIMULATED KEYSTROKES
                                        ;       THIS TABLE REPRESENTS A 4*2 ARRAY.  EACH ROW
                                        ;       CONSISTS OF A SEQUENCE OF SCAN CODES WHICH
                                        ;       WOULD HAVE BEEN GENERATED ON AN 83 KEY KEYBOARD
                                        ;       TO CAUSE THE FOLLOWING FUNCTIONS:
                                        ;               ROW 1=ECHO CRT OUTPUT TO THE PRINTER
                                        ;               ROW 2=BREAK
                                        ;       THE TABLE HAS BOTH MAKE AND BREAK SCAN CODES.
                                        ;----------------------------------------------
 108B                                   SCAN    LABEL BYTE
 108B  1D 37 B7 9D                              DB 29,55,183,157         ; CTRL + PRTSC
 108F  1D 46 C6 9D                              DB 29,70,198,157         ; CTRL + SCROLL-LOCK
```

```
                                ;-------------------------------------------------
                                ;TABLE OF VALID ALT SHIFT SCAN CODES
                                ;       THIS TABLE CONTAINS SCAN CODES FOR KEYS ON THE
                                ;       62 KEY KEYBOARD.  THESE CODES ARE USED IN
                                ;       COMBINATION WITH THE ALT KEY TO PRODUCE SCAN CODES
                                ;       FOR KEYS NOT FOUND ON THE 62 KEY KEYBOARD.
                                ;-------------------------------------------------
1093                            ALT_TABLE       LABEL   BYTE
1093  35 28 34 1A 1B                    DB 53,40,52,26,27
= 0005                          ALT_LEN  EQU $ - ALT_TABLE
                                ;-------------------------------------------------
                                ;TABLE OF TRANSLATED SCAN CODES WITH ALT SHIFT
                                ;       THIS TABLE CONTAINS THE SCAN CODES FOR THE
                                ;       KEYS WHICH ARE NOT ON THE 62 KEY KEYBOARD AND
                                ;       WILL BE TRANSLATED WITH ALT SHIFT.  THERE IS A
                                ;       ONE TO ONE CORRESPONDENCE BETWEEN THE SIZES
                                ;       OF ALT_TABLE AND NEW_ALT.
                                ;       THE FOLLOWING TRANSLATIONS ARE MADE:
                                ;               ALT+ / = \
                                ;               ALT+ ' = `
                                ;               ALT+ [ = ¦
                                ;               ALT+ ] = ~
                                ;               ALT+ . = *
                                ;-------------------------------------------------
1098                            NEW_ALT LABEL BYTE
1098  2B 29 37 2B 29                    DB 43,41,55,43,41
                                ;-------------------------------------------------
                                ;EXTAB
                                ;       TABLE OF SCAN CODES FOR MAPPING EXTENDED SET
                                ;       OF SCAN CODES (SCAN CODES > 85).  THIS TABLE
                                ;       ALLOWS OTHER DEVICES TO USE THE KEYBOARD INTERFACE.
                                ;       IF THE DEVICE GENERATES A SCAN CODE > 85 THIS TABLE
                                ;       CAN BE USED TO MAP THE DEVICE TO THE KEYBOARD.  THE
                                ;       DEVICE ALSO HAS THE OPTION OF HAVING A UNIQUE SCAN
                                ;       CODE PUT IN THE KEYBOARD BUFFER (INSTEAD OF MAPPING
                                ;       TO THE KEYBOARD). THE EXTENDED SCAN CODE PUT IN THE
                                ;       BUFFER WILL BE CONTINUOUS BEGINNING AT 150.  A ZERO
                                ;       WILL BE USED IN PLACE OF AN ASCII CODE. (E.G. A
                                ;       DEVICE GENERATING SCAN CODE 86 AND NOT MAPPING 86
                                ;       TO THE KEYBOARD WILL HAVE A [150,0] PUT IN THE
                                ;       KEYBOARD BUFFER)
                                ;       TABLE FORMAT:
                                ;       THE FIRST BYTE IS A LENGTH INDICATING THE NUMBER
                                ;       OF SCAN CODES MAPPED TO THE KEYBOARD.  THE REMAINING
                                ;       ENTRIES ARE WORDS. THE FIRST BYTE (LOW BYTE) IS A
                                ;       SCAN CODE AND THE SECOND BYTE (HIGH BYTE) IS ZERO.
                                ;       A DEVICE GENERATING N SCAN CODES IS ASSUMED TO GENERATE THE
                                ;       FOLLOWING STREAM 86,87,88,...,86+(N-1). THE SCAN CODE BYTES
                                ;       IN THE TABLE CORRESPOND TO THIS SET WITH THE FIRST DATA
                                ;       BYTE MATCHING 86, THE SECOND MATCHING 87 ETC.
                                ;   NOTES:
                                ;       (1) IF A DEVICE GENERATES A BREAK CODE, NOTHING IS
                                ;           PUT IN THE BUFFER.
                                ;       (2) A LENGTH OF 0 INDICATES THAT ZERO SCAN CODES HAVE BEEN
                                ;           MAPPED TO THE KEYBOARD AND ALL EXTENDED SCAN CODES WILL
                                ;           BE USED.
                                ;       (3) A DEVICE CAN MAP SOME OF ITS SCAN CODES TO THE KEYBOARD
                                ;           AND HAVE SOME ITS SCAN CODES IN THE EXTENDED SET.
                                ;-------------------------------------------------
109D                            EXTAB   LABEL BYTE
109D  14                                DB      20                ; LENGTH OF TABLE
109E  0048 0049 004D 0051               DW      72,73,77,81,80,79,75,71,57,28
      0050 004F 004B 0047
      0039 001C
10B2  0011 0012 001F 002D               DW      17,18,31,45,44,43,30,16,15,1
      002C 002B 001E 0010
      000F 0001
10C6                            KEY62_INT PROC FAR
10C6  FB                                 STI
10C7  FC                                 CLD                      ; FORWARD DIRECTION
10C8  E8 138B R                          CALL   DDS               ; SET UP ADDRESSING
10CB  8A E0                              MOV    AH,AL             ; SAVE SCAN CODE
10CD  E8 131E R                          CALL   TPM               ; ADJUST OUTPUT FOR USER
                                                                  ; MODIFICATION
10D0  73 01                              JNC    KBXO              ; JUMP IF OK TO CONTINUE
10D2  CF                                 IRET                     ; RETURN FROM INTERRUPT.
                                ;----EXTENDED SCAN CODE CHECK
10D3  3C FF                     KBXO:   CMP     AL,OFFH           ; IS THIS AN OVERRUN CHAR?
10D5  74 6C                             JE      KBO_1             ; PASS IT TO INTERRUPT 9
10D7  24 7F                             AND     AL,AND_MASK-BREAK_BIT ; TURN OFF BREAK BIT
10D9  3C 56                             CMP     AL,EXT_SCAN       ; IS THIS A SCAN CODE > 83
10DB  7C 5F                             JL      KBX4              ; REPLACE BREAK BIT
                                ;----SCAN CODE IS IN EXTENDED SET
10DD  1E                                PUSH    DS
10DE  33 F6                             XOR     SI,SI
10E0  8E DE                             MOV     DS,SI
                                        ASSUME  DS:ABSO
10E2  C4 3E 0124 R                      LES     DI,DWORD PTR EXST ; GET THE POINTER TO THE EXTENDED
                                                                  ; SET
10E6  26: 8A 0D                         MOV     CL,BYTE PTR ES:[DI] ; GET LENGTH BYTE
10E9  1F                                POP     DS
                                        ASSUME  DS:DATA
                                ;----DOES SCAN CODE GET MAPPED TO KEYBOARD OR TO NEW EXTENDED SCAN
                                ;     CODES?
10EA  2C 56                             SUB     AL,EXT_SCAN       ; CONVERT TO BASE OF NEW SET
10EC  FE C9                             DEC     CL                ; LENGTH - 1
10EE  3A C1                             CMP     AL,CL             ; IS CODE IN TABLE?
10F0  7F 10                             JG      KBX1              ; JUMP IF SCAN CODE IS NOT IN TABLE
```

```
                              ;----GET SCAN CODE FROM TABLE
10F2  47                              INC    DI                ; POINT DI PAST LENGTH BYTE
10F3  8B D8                           MOV    BX,AX
10F5  32 FF                           XOR    BH,BH             ; PREPARE FOR ADDING TO 16 BIT
                                                               ; REGISTER
10F7  D1 E3                           SHL    BX,1
10F9  03 FB                           ADD    DI,BX             ; OFFSET TO CORRECT TABLE ENTRY
10FB  26: 8A 05                       MOV    AL,BYTE PTR ES:[DI] ; TRANSLATED SCAN CODE IN AL
10FE  3C 56                           CMP    AL,EXT_SCAN       ; IS CODE IN KEYBOARD SET?
1100  7C 3A                           JL     KBX4              ; IN KEYBOARD SET, CHECK FOR BREAK
                              ;----SCAN CODE GETS MAPPED TO EXTENDED SCAN CODES
1102  F6 C4 80                KBX1:   TEST   AH,BREAK_BIT      ; IS THIS A BREAK CODE?
1105  74 01                           JZ     KBX2              ; MAKE CODE, PUT IN BUFFER
1107  CF                              IRET                     ; BREAK CODE, RETURN FROM INTERRUPT
1108  80 C4 40                KBX2:   ADD    AH,64             ; EXTENDED SET CODES BEGIN AT 150
110B  32 C0                           XOR    AL,AL             ; ZERO OUT ASCII VALUE (NUL)
110D  8B 1E 001C R                    MOV    BX,BUFFER_TAIL    ; GET TAIL POINTER
1111  8B F3                           MOV    SI,BX             ; SAVE POINTER TO TAIL
1113  E8 144F R                       CALL   K4                ; INCREMENT TAIL VALUE
1116  3B 1E 001A R                    CMP    BX,BUFFER_HEAD    ; IS BUFFER FULL?
111A  75 19                           JNE    KBX3              ; PUT CONTENTS OF AX IN BUFFER
                              ;----BUFFER IS FULL, BEEP AND CLEAR FLAGS
111C  BB 0080                         MOV    BX,80H            ; FREQUENCY OF BEEP
111F  B9 0048                         MOV    CX,48H            ; DURATION OF BEEP
1122  E8 E035 R                       CALL   KB_NOISE          ; BUFFER FULL BEEP
1125  80 26 0017 R F0                 AND    KB_FLAG,OFOH      ; CLEAR ALT, CTRL, LEFT AND RIGHT
                                                               ; SHIFTS
112A  80 26 0018 R OF                 AND    KB_FLAG_1,OFH     ; CLEAR MAKE OF INS,CAPS_LOCK,NUM
                                                               ; AND SCROLL
112F  80 26 0088 R 1F                 AND    KB_FLAG_2,1FH     ; CLEAR FUNCTION STATES
1134  CF                              IRET                     ; DONE WITH INTERRUPT
1135  89 04                   KBX3:   MOV    [SI],AX           ; PUT CONTENTS OF AX IN BUFFER
1137  89 1E 001C R                    MOV    BUFFER_TAIL,BX    ; ADVANCE BUFFER TAIL
113B  CF                              IRET                     ; RETURN FROM INTERRUPT
113C  80 E4 80                KBX4:   AND    AH,BREAK_BIT      ; MASK BREAK BIT ON ORIGINAL SCAN
113F  0A C4                           OR     AL,AH             ; UPDATE NEW SCAN CODE
1141  8A E0                           MOV    AH,AL             ; SAVE AL IN AH AGAIN
                              ;----83 KEY KEYBOARD FUNCTIONS SHIFT+PRTSC AND CTRL+NUMLOCK
1143  3C 45                   KB0_1:  CMP    AL,NUM_KEY        ; IS THIS A NUMLOCK?
1145  75 14                           JNE    KB0_3             ; CHECK FOR PRTSC
1147  F6 06 0017 R 04                 TEST   KB_FLAG,CTL_SHIFT ; IS CTRL KEY BEING HELD DOWN?
114C  74 0A                           JZ     KB0_2             ; NUMLOCK WITHOUT CTRL, CONTINUE
114E  F6 06 0017 R 08                 TEST   KB_FLAG,ALT_SHIFT ; IS ALT KEY HELD CONCURRENTLY?
1153  75 03                           JNZ    KB0_2             ; PASS IT ON
1155  E9 12EB R                       JMP    KB16_1            ; PUT KEYBOARD IN HOLD STATE
1158  E9 125C R               KB0_2:  JMP    CONT_INT          ; CONTINUE WITH INTERRUPT 48H
                              ;----CHECK FOR PRTSC
115B  3C 37                   KB0_3:  CMP    AL,55             ; IS THIS A PRTSC KEY?
115D  75 11                           JNZ    KB1_1             ; NOT A PRTSC KEY
115F  F6 06 0017 R 03                 TEST   KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; EITHER SHIFT
                                                               ; ACTIVE?
1164  74 F2                           JZ     KB0_2             ; PROCESS SCAN IN INT9
1166  F6 06 0017 R 04                 TEST   KB_FLAG,CTL_SHIFT ; IS THE CTRL KEY PRESSED?
116B  75 EB                           JNZ    KB0_2             ; NOT A VALID PRTSC (PC COMPATIBLE)
116D  E9 1301 R                       JMP    PRTSC             ; HANDLE THE PRINT SCREEN FUNCTION
                              ;----ALTERNATE SHIFT TRANSLATIONS
1170  8A E0                   KB1_1:  MOV    AH, AL            ; SAVE CHARACTER
1172  24 7F                           AND    AL, AND_MASK - BREAK_BIT ; MASK BREAK BIT
1174  F6 06 0017 R 08                 TEST   KB_FLAG,ALT_SHIFT ; IS THIS A POTENTIAL TRANSLATION
1179  74 39                           JZ     KB2
                              ;----TABLE LOOK UP
117B  0E                              PUSH   CS
117C  07                              POP    ES                ; INITIALIZE SEGMENT FOR TABLE LOOK
                                                               ; UP
117D  BF 1093 R                       MOV    DI,OFFSET ALT_TABLE
1180  B9 0005                         MOV    CX,ALT_LEN        ; GET READY FOR TABLE LOOK UP
1183  F2/ AE                          REPNE  SCASB             ; SERACH TABLE
1185  75 2D                           JNE    KB2               ; JUMP IF MATCH IS NOT FOUND
1187  B9 1094 R                       MOV    CX,OFFSET ALT_TABLE + 1
118A  2B F9                           SUB    DI,CX             ; UPDATE DI TO INDEX SCAN CODE
118C  2E: 8A 85 1098 R                MOV    AL,CS:NEW_ALT[DI] ; TRANSLATE SCAN CODE
                              ;----CHECK FOR BREAK CODE
1191  8A 1E 0017 R                    MOV    BL,KB_FLAG        ; SAVE KB_FLAG STATUS
1195  80 36 0017 R 08                 XOR    KB_FLAG,ALT_SHIFT ; MASK OFF ALT SHIFT
119A  F6 C4 80                        TEST   AH,BREAK_BIT      ; IS THIS A BREAK CHARACTER?
119D  74 02                           JZ     KB1_2             ; JUMP IF SCAN IS A MAKE
119F  0C 80                           OR     AL,BREAK_BIT      ; SET BREAK BIT
                              ;----MAKE CODE, CHECK FOR SHIFT SEQUENCE
11A1  83 FF 03                KB1_2:  CMP    DI,3              ; IS THIS A SHIFT SEQUENCE
11A4  7C 05                           JL     KB1_3             ; JUMP IF NOT SHIFT SEQUENCE
11A6  80 0E 0017 R 02                 OR     KB_FLAG,LEFT_SHIFT ; TURN ON SHIFT FLAG
11AB  E6 60                   KB1_3:  OUT    KBPORT,AL
11AD  CD 09                           INT    9H                ; ISSUE INT TO PROCESS SCAN CODE
11AF  88 1E 0017 R                    MOV    KB_FLAG,BL        ; RESTORE ORIGINAL FLAG STATES
11B3  CF                              IRET
                              ;----FUNCTION KEY HANDLER
11B4  3C 54                   KB2:    CMP    AL, FN_KEY        ; CHECK FOR FUNCTION KEY
11B6  75 23                           JNZ    KB4               ; JUMP IF NOT FUNCTION KEY
11B8  F6 C4 80                        TEST   AH, BREAK_BIT     ; IS THIS A FUNCTION BREAK
11BB  75 0B                           JNZ    KB3               ; JUMP IF FUNCTION BREAK
11BD  80 26 0088 R 1F                 AND    KB_FLAG_2,CLEAR_FLAGS ; CLEAR ALL PREVIOUS
                                                               ; FUNCTIONS
11C2  80 0E 0088 R A0                 OR     KB_FLAG_2, FN_FLAG + FN_PENDING
11C7  CF                              IRET                     ; RETURN FROM INTERRUPT
                              ;----FUNCTION BREAK
11C8  F6 06 0088 R 20         KB3:    TEST   KB_FLAG_2,FN_PENDING
11CD  75 06                           JNZ    KB3_1             ; JUMP IF FUNCTION IS PENDING
11CF  80 26 0088 R 1F                 AND    KB_FLAG_2,CLEAR_FLAGS ; CLEAR ALL FLAGS
11D4  CF                              IRET
11D5  80 0E 0088 R 40         KB3_1:  OR     KB_FLAG_2,FN_BREAK ; SET BREAK FLAG
11DA  CF                      KB3_2:  IRET                     ; RETURN FROM INTERRUPT
```

```
                                ;----CHECK IF FUNCTION FLAG ALREADY SET
11DB  3C 55                     KB4:    CMP      AL,PHK            ; IS THIS A PHANTOM KEY?
11DD  74 FB                             JZ       KB3_2             ; JUMP IF PHANTOM SEQUENCE
11DF  F6 06 0088 R 90           KB4_0:  TEST     KB_FLAG_2,FN_FLAG+FN_LOCK ; ARE WE IN FUNCTION
                                                                   ; STATE?
11E4  75 21                             JNZ      KB5
                                ;----CHECK IF NUM_STATE IS ACTIVE
11E6  F6 06 0017 R 20                   TEST     KB_FLAG,NUM_STATE
11EB  74 16                             JZ       KB4_1             ; JUMP IF NOT IN NUM_STATE
11ED  3C 0B                             CMP      AL,NUM_0          ; ARE WE IN NUMERIC KEYPAD REGION?
11EF  77 12                             JA       KB4_1             ; JUMP IF NOT IN KEYPAD
11F1  FE C8                             DEC      AL                ; CHECK LOWER BOUND OF RANGE
11F3  74 0E                             JZ       KB4_1             ; JUMP IF NOT IN RANGE (ESC KEY)
                                ;----TRANSLATE SCAN CODE TO NUMERIC KEYPAD
11F5  FE C8                             DEC      AL                ; AL IS OFFSET INTO TABLE
11F7  BB 1081 R                         MOV      BX,OFFSET NUM_CODES
11FA  2E: D7                            XLAT     CS:NUM_CODES      ; NEW SCAN CODE IS IN AL
11FC  80 E4 80                          AND      AH,BREAK_BIT      ; ISOLATE BREAK BIT ON ORIGINAL
                                                                   ; SCAN CODE
11FF  0A C4                             OR       AL,AH             ; UPDATE KEYPAD SCAN CODE
1201  EB 59                             JMP      SHORT CONT_INT    ; CONTINUE WITH INTERRUPT
1203  8A C4                     KB4_1:  MOV      AL,AH             ; GET BACK BREAK BIT IF SET
1205  EB 55                             JMP      SHORT CONT_INT
                                ;----CHECK FOR VALID FUNCTION KEY
1207  3C 0B                     KB5:    CMP      AL, NUM_0         ; CHECK FOR RANGE OF INTEGERS
1209  77 2D                             JA       KB7               ; JUMP IF NOT IN RANGE
120B  FE C8                             DEC      AL                ; CHECK FOR ESC KEY (=1)
120D  75 25                             JNZ      KB6               ; NOT ESCAPE KEY, RANGE OF INTEGERS
                                ;----ESCAPE KEY, LOCK KEYBOARD IN FUNCTION LOCK
120F  F6 C4 80                          TEST     AH,BREAK_BIT      ; IS THIS A BREAK CODE?
1212  75 30                             JNZ      KB8               ; NO PROCESSING FOR ESCAPE BREAK
1214  F6 06 0088 R 80                   TEST     KB_FLAG_2,FN_FLAG ; TOGGLES ONLY WHEN FN HELD
                                                                   ; CONCURRENTLY
1219  74 29                             JZ       KB8               ; NOT HELD CONCURRENTLY
121B  F6 06 0088 R 40                   TEST     KB_FLAG_2,FN_BREAK ; HAS THE FUNCTION KEY BEEN
                                                                   ; RELEASED?
1220  75 22                             JNZ      KB8               ; CONTINUE IF RELEASED.  PROCESS AS
                                                                   ; ESC
1222  F6 06 0017 R 03                   TEST     KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; EITHER SHIFT?
1227  74 1B                             JZ       KB8               ; NOT HELD DOWN
1229  80 36 0088 R 10                   XOR      KB_FLAG_2,FN_LOCK ; TOGGLE STATE
122E  80 26 0088 R 1F                   AND      KB_FLAG_2,CLEAR_FLAGS ; TURN OFF OTHER STATES
1233  CF                                IRET                       ; RETURN FROM INTERUPT
                                ;----SCAN CODE IN RANGE 1 -> 0
1234  04 3A                     KB6:    ADD      AL, 58            ; GENERATE CORRECT SCAN CODE
1236  EB 3E                             JMP      SHORT KB12        ; CLEAN-UP BEFORE RETURN TO KB_INT
                                ;----CHECK TABLE FOR OTHER VALID SCAN CODES
1238  0E                        KB7:    PUSH     CS
1239  07                                POP      ES                ; ESTABLISH ADDRESS OF TABLE
123A  BF 1069 R                         MOV      DI, OFFSET KB0    ; BASE OF TABLE
123D  B9 000C                           MOV      CX, KB0LEN        ; LENGTH OF TABLE
1240  F2/ AE                            REPNE    SCASB             ; SEARCH TABLE FOR A MATCH
1242  74 1D                             JE       KB10              ; JUMP IF MATCH
                                ;----ILLEGAL CHARACTER
1244  F6 06 0088 R 40           KB8:    TEST     KB_FLAG_2,FN_BREAK ; HAS BREAK OCCURED?
1249  74 0F                             JZ       KB9               ; FUNCTION KEY HAS NOT BEEN
                                                                   ; RELEASED
124B  F6 C4 80                          TEST     AH,BREAK_BIT      ; IS THIS A BREAK OF AN ILLEGAL
124E  75 0A                             JNZ      KB9               ; DON'T RESET FLAGS ON ILLEGAL
                                                                   ; BREAK
1250  80 26 0088 R 1F           KB85:   AND      KB_FLAG_2,CLEAR_FLAGS ; NORMAL STATE
1255  C6 06 0087 R 00                   MOV      CUR_FUNC,0        ; RETRIEVE ORIGINAL SCAN CODE
                                ;----FUNCTION BREAK IS NOT SET
125A  8A C4                     KB9:    MOV      AL,AH             ; RETRIEVE ORIGINAL SCAN CODE
125C                            CONT_INT:
125C  E6 60                             OUT      KBPORT,AL
125E  CD 09                             INT      9H                ; ISSUE KEYBOARD INTERRUPT
1260                            RET_INT:
1260  CF                                IRET
                                ;----BEFORE TRANSLATION CHECK FOR ALT+FN+N_KEY AS NUM LOCK
1261  3C 31                     KB10:   CMP      AL,N_KEY          ; IS THIS A POTENTIAL NUMLOCK?
1263  75 07                             JNE      KB10_1            ; NOT A NUMKEY, TRANSLATE IT
1265  F6 06 0017 R 08                   TEST     KB_FLAG,ALT_SHIFT ; ALT HELD DOWN ALSO?
126A  74 D8                             JZ       KB8               ; TREAT AS ILLEGAL COMBINATION
126C  B9 106A R                 KB10_1: MOV      CX, OFFSET KB0 + 1 ; GET OFFSET TO TABLE
126F  2B F9                             SUB      DI, CX            ; UPDATE INDEX TO NEW SCAN CODE
                                                                   ; TABLE
1271  2E: 8A 85 1075 R                  MOV      AL, CS:KB1[DI]    ; MOV NEW SCAN CODE INTO REGISTER
                                ;----TRANSLATED CODE IN AL OR AN OFFSET TO THE TABLE "SCAN"
1276  F6 C4 80                  KB12:   TEST     AH,BREAK_BIT      ; IS THIS A BREAK CHAR?
1279  74 35                             JZ       KB13              ; JUMP IF MAKE CODE
                                ;----CHECK FOR TOGGLE KEY
127B  3C 45                             CMP      AL,NUM_LOCK       ; IS THIS A NUM LOCK?
127D  74 04                             JZ       KB12_1            ; JUMP IF TOGGLE KEY
127F  3C 46                             CMP      AL,SCROLL_LOCK    ; IS THIS A SCROLL LOCK?
1281  75 08                             JNZ      KB12_2            ; JUMP IF NOT A TOGGLE KEY
1283  0C 80                     KB12_1: OR       AL,80H            ; TURN ON BREAK BIT
1285  E6 60                             OUT      KBPORT,AL
1287  CD 09                             INT      9H                ; TOGGLE STATE
1289  24 7F                             AND      AL,AND_MASK-BREAK_BIT ; TURN OFF BREAK BIT
128B  F6 06 0088 R 40           KB12_2: TEST     KB_FLAG_2,FN_BREAK ; HAS FUNCTION BREAK OCCURED?
1290  74 11                             JZ       KB12_3            ; JUMP IF BREAK HAS NOT OCCURED
1292  3A 06 0087 R                      CMP      AL,CUR_FUNC       ; IS THIS A BREAK OF OLD VALID
                                                                   ; FUNCTION
1296  75 C8                             JNE      RET_INT           ; ALLOW FURTHER CURRENT FUNCTIONS
1298  80 26 0088 R 1F                   AND      KB_FLAG_2,CLEAR_FLAGS
129D                            KB12_20:
129D  C6 06 0087 R 00                   MOV      CUR_FUNC,0        ; CLEAR CURRENT FUNCTION
12A2  CF                                IRET                       ; RETURN FROM INTERRUPT
```

```
12A3  3A 06 0087 R          KB12_3: CMP     AL,CUR_FUNC       ; IS THIS BREAK OF FIRST FUNCTION?
12A7  75 B7                         JNE     RET_INT           ; IGNORE
12A9  80 26 0088 R DF               AND     KB_FLAG_2,AND_MASK-FN_PENDING ; TURN OFF PENDING
                                                              ; FUNCTION
12AE  EB ED                         JMP     KB12_20           ; CLEAR CURRENT FUNCTION AND RETURN
                            ;----VALID MAKE KEY HAS BEEN PRESSED
12B0  F6 06 0088 R 40       KB13:   TEST    KB_FLAG_2,FN_BREAK ; CHECK IF FUNCTION KEY HAS BEEN
                                                              ; PRESSED
12B5  74 0D                         JZ      KB14_1            ; JUMP IF NOT SET
                            ;----FUNCTION BREAK HAS ALREADY OCCURED
12B7  80 3E 0087 R 00               CMP     CUR_FUNC,0        ; IS THIS A NEW FUNCTION?
12BC  74 06                         JZ      KB14_1            ; INITIALIZE NEW FUNCTION
12BE  38 06 0087 R                  CMP     CUR_FUNC,AL       ; IS THIS NON-CURRENT FUNCTION
12C2  75 8C                         JNZ     KB85              ;JUMP IF NO FUNCTION IS PENDING
                                                              ; ..TO RETRIEVE ORIGINAL SCAN CODE
                            ;----CHECK FOR SCAN CODE GENERATION SEQUENCE
12C4  A2 0087 R             KB14_1: MOV     CUR_FUNC,AL       ; INITIALIZE CURRENT FN
12C7  3C 04                 KB16:   CMP     AL,PRT_SCREEN     ; IS THIS A SIMULATED SEQUENCE?
12C9  7F 91                         JG      CONT_INT          ; JUMP IF THIS IS A SIMPLE
                                                              ; TRANSLATION
12CB  74 34                         JZ      PRTSC             ; DO THE PRINT SCREEN FUNCTION
12CD  3C 03                         CMP     AL,PAUSE          ; IS THIS THE HOLD FUNCTION?
12CF  74 1A                         JZ      KB16_1            ; DO THE PAUSE FUNCTION
                            ;----BREAK OR ECHO
12D1  FE C8                         DEC     AL                ; POINT AT BASE
12D3  D0 E0                         SHL     AL,1
12D5  D0 E0                         SHL     AL,1              ; MULTIPLY BY 4
12D7  98                            CBW
12D8  2E: 8D 36 1088 R              LEA     SI,SCAN           ; ADDRESS SEQUENCE OF SIMULATED
                                                              ; KEYSTROKES
12DD  03 F0                         ADD     SI,AX             ; UPDATE TO POINT AT CORRECT SET
12DF  B9 0004                       MOV     CX,4              ; LOOP COUNTER
12E2                        GENERATE:
12E2  2E: AC                        LODS    SCAN              ; GET SCAN CODE FROM TABLE
12E4  E6 60                         OUT     KBPORT,AL
12E6  CD 09                         INT     9H                ; PROCESS IT
12E8  E2 F8                         LOOP    GENERATE          ; GET NEXT
12EA  CF                            IRET
                            ;----PUT KEYBOARD IN HOLD STATE
12EB  F6 06 0018 R 08       KB16_1: TEST    KB_FLAG_1,HOLD_STATE ; CANNOT GO IN HOLD STATE IF
                                                              ; ITS ACTIVE
12F0  75 0E                         JNZ     KB16_2            ; DONE WITH INTERRUPT
12F2  80 0E 0018 R 08               OR      KB_FLAG_1,HOLD_STATE ; TURN ON HOLD FLAG
12F7  E4 A0                         IN      AL,NMI_PORT       ; RESET KEYBOARD LATCH
12F9  F6 06 0018 R 08       HOLD:   TEST    KB_FLAG_1,HOLD_STATE ; STILL IN HOLD STATE?
12FE  75 F9                         JNZ     HOLD              ; CONTINUE LOOPING UNTIL KEY IS
                                                              ; PRESSED
1300  CF                    KB16_2: IRET                      ; RETURN FROM INTERRUPT 48H
                            ;----PRINT SCREEN FUNCTION
1301  F6 06 0018 R 08       PRTSC:  TEST    KB_FLAG_1,HOLD_STATE ; IS HOLD STATE IN PROGRESS?
1306  74 06                         JZ      KB16_3            ; OK TO CONTINUE WITH PRTSC
1308  80 26 0018 R F7               AND     KB_FLAG_1,0FFH-HOLD_STATE ; TURN OFF FLAG
130D  CF                            IRET
130E  83 C4 06              KB16_3: ADD     SP,3*2            ; GET RID OF CALL TO INTERRUPT 48H
1311  07                            POP     ES                ; POP REGISTERS THAT AREN'T
                                                              ; MODIFIED IN INT5
1312  1F                            POP     DS
1313  5A                            POP     DX
1314  59                            POP     CX
1315  5B                            POP     BX
1316  E4 A0                         IN      AL,NMI_PORT       ; RESET KEYBOARD LATCH
1318  CD 05                         INT     5H                ; ISSUE INTERRUPT
131A  58                            POP     AX
131B  5F                            POP     DI
131C  5E                            POP     SI                ; POP THE REST
131D  CF                            IRET
131E                        KEY62_INT ENDP
                            ;-------------------------------------------------------------------
                            ;TYPAMATIC
                            ;       THIS ROUTINE WILL CHECK KEYBOARD STATUS BITS IN KB_FLAG_2
                            ;       AND DETERMINE WHAT STATE THE KEYBOARD IS IN.  APPROPRIATE
                            ;       ACTION WILL BE TAKEN.
                            ;INPUT
                            ;       AL= SCAN CODE OF KEY WHICH TRIGGERED NON-MASKABLE INTERRUPT
                            ;OUTPUT
                            ;       CARRY BIT = 1 IF NO ACTION IS TO BE TAKEN.
                            ;       CARRY BIT = 0 MEANS SCAN CODE IN AL SHOULD BE PROCESSED
                            ;                 FURTHER.
                            ;       MODIFICATIONS TO THE VARIABLES CUR_CHAR AND VAR_DELAY ARE
                            ;       MADE.  ALSO THE PUTCHAR BIT IN KB_FLAG_2 IS TOGGLED WHEN
                            ;       THE KEYBOARD IS IN HALF RATE MODE.
                            ;-------------------------------------------------------------------
131E                        TPM     PROC    NEAR
131E  53                            PUSH    BX
131F  38 06 0085 R                  CMP     CUR_CHAR,AL       ; IS THIS A NEW CHARACTER?
1323  74 31                         JZ      TP2               ; JUMP IF SAME CHARACTER
                            ;----NEW CHARACTER CHECK FOR BREAK SEQUENCES
1325  A8 80                         TEST    AL,BREAK_BIT      ; IS THE NEW KEY A BREAK KEY?
1327  74 12                         JZ      TP0               ; JUMP IF NOT A BREAK
1329  24 7F                         AND     AL,07FH           ; CLEAR BREAK BIT
132B  38 06 0085 R                  CMP     CUR_CHAR,AL       ; IS NEW CHARACTER THE BREAK OF
                                                              ; LAST MAKE?
132F  8A C4                         MOV     AL,AH             ; RETRIEVE ORIGINAL CHARACTER
1331  75 05                         JNZ     TP                ; JUMP IF NOT THE SAME CHARACTER
1333  C6 06 0085 R 00               MOV     CUR_CHAR,00       ; CLEAR CURRENT CHARACTER
1338  F8                    TP:     CLC                       ; CLEAR CARRY BIT
1339  5B                            POP     BX
133A  C3                            RET                       ; RETURN
```

```
                                    ;----INITIALIZE A NEW CHARACTER
133B  A2 0085 R                     TP0:    MOV     CUR_CHAR,AL       ; SAVE NEW CHARACTER
133E  80 26 0086 R F0                       AND     VAR_DELAY,0F0H    ; CLEAR VARIABLE DELAY
1343  80 26 0088 R FE                       AND     KB_FLAG_2,0FEH    ; INITIAL PUTCHAR BIT AS ZERO
1348  F6 06 0088 R 02                       TEST    KB_FLAG_2,INIT_DELAY ; ARE WE INCREASING THE
                                                                      ; INITIAL DELAY?
134D  74 E9                                 JZ      TP                ; DEFAULT DELAY
134F  80 0E 0086 R 0F                       OR      VAR_DELAY,DELAY_RATE ; INCREASE DELAY BY 2X
1354  EB E2                                 JMP     SHORT TP
                                    ;----CHECK IF WE ARE IN TYPAMATIC MODE AND IF DELAY IS OVER
1356  F6 06 0088 R 08               TP2:    TEST    KB_FLAG_2,TYPE_OFF ; IS TYPAMATIC TURNED OFF?
135B  75 2B                                 JNZ     TP4               ; JUMP IF TYPAMATIC RATE IS OFF
135D  8A 1E 0086 R                          MOV     BL,VAR_DELAY      ; GET VAR_DEALY
1361  80 E3 0F                              AND     BL,0FH            ; MASK OFF HIGH ORDER(SCREEN RANGE)
1364  0A DB                                 OR      BL,BL             ; IS INITIAL DELAY OVER?
1366  74 0D                                 JZ      TP3               ; JUMP IF DELAY IS OVER
1368  FE CB                                 DEC     BL                ; DECREASE DELAY WAIT BY ANOTHER
                                                                      ; CHARACTER
136A  80 26 0086 R F0                       AND     VAR_DELAY,0F0H
136F  08 1E 0086 R                          OR      VAR_DELAY,BL
1373  EB 13                                 JMP     SHORT TP4
                                    ;----CHECK IF TIME TO OUTPUT CHAR
1375  F6 06 0088 R 04               TP3:    TEST    KB_FLAG_2,HALF_RATE ; ARE WE IN HALF RATE MODE
137A  74 BC                                 JZ      TP                ; JUMP IF WE ARE IN NORMAL MODE
137C  80 36 0088 R 01                       XOR     KB_FLAG_2,PUTCHAR ; TOGGLE BIT
1381  F6 06 0088 R 01                       TEST    KB_FLAG_2,PUTCHAR ; IS IT TIME TO PUT OUT A CHAR
1386  75 B0                                 JNZ     TP                ; NOT TIME TO OUTPUT CHARACTER
1388                                TP4:                              ; SKIP THIS CHARACTER
1388  F9                                    STC                       ; SET CARRY FLAG
1389  5B                                    POP     BX
138A  C3                                    RET
138B                                TPM     ENDP
                                    ;--------------------------------------------------------------
                                    ; THIS SUBROUTINE SETS DS TO POINT TO THE BIOS DATA AREA
                                    ; INPUT: NONE
                                    ; OUTPUT: DS IS SET
                                    ;--------------------------------------------------------------
138B                                DDS     PROC    NEAR
138B  50                                    PUSH    AX
138C  B8 0040                               MOV     AX,40H
138F  8E D8                                 MOV     DS,AX
1391  58                                    POP     AX
1392  C3                                    RET
1393                                DDS     ENDP
                                    ;--- INT 1A ---------------------------------------------------
                                    ; TIME_OF_DAY/SOUND SOURCE SELECT
                                    ;       THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ.
                                    ;       AN INTERFACE FOR SETTING THE MULTIPLEXER FOR
                                    ;       AUDIO SOURCE IS ALSO PROVIDED
                                    ;
                                    ; INPUT
                                    ;   (AH) = 0    READ THE CURRENT CLOCK SETTING
                                    ;               RETURNS CX = HIGH PORTION OF COUNT
                                    ;                       DX = LOW PORTION OF COUNT
                                    ;                       AL = 0 IF TIMER HAS NOT PASSED 24 HOURS
                                    ;                       SINCE LAST READ. <> 0 IF ON ANOTHER DAY
                                    ;   (AH) = 1    SET THE CURRENT CLOCK
                                    ;       CX = HIGH PORTION OF COUNT
                                    ;       DX = LOW PORTION OF COUNT
                                    ;   (AH) = 80H  SET UP SOUND MULTIPLEXER
                                    ;       AL =(SOURCE OF SOUND) --> "AUDIO OUT" OR RF MODULATOR
                                    ;            00 = 8253 CHANNEL 2
                                    ;            01 = CASSETTE INPUT
                                    ;            02 = "AUDIO IN" LINE ON I/O CHANNEL
                                    ;            03 = COMPLEX SOUND GENERATOR CHIP
                                    ;
                                    ; NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SEC
                                    ;       (OR ABOUT 18.2 PER SECOND -- SEE EQUATES BELOW)
                                    ;--------------------------------------------------------------
                                            ASSUME  CS:CODE,DS:DATA
1393                                TIME_OF_DAY     PROC    FAR
1393  FB                                    STI                       ; INTERRUPTS BACK ON
1394  1E                                    PUSH    DS                ; SAVE SEGMENT
1395  E8 138B R                             CALL    DDS
1398  80 FC 80                              CMP     AH,80H            ; AH=80
139B  74 2E                                 JE      T4A               ; MUX_SET-UP
139D  0A E4                                 OR      AH,AH             ; AH=0
139F  74 07                                 JZ      T2                ; READ_TIME
13A1  FE CC                                 DEC     AH                ; AH=1
13A3  74 16                                 JZ      T3                ; SET_TIME
13A5  FB                            T1:     STI                       ; INTERRUPTS BACK ON
13A6  1F                                    POP     DS                ; RECOVER SEGMENT
13A7  CF                                    IRET                      ; RETURN TO CALLER
13A8  FA                            T2:     CLI                       ; NO TIMER INTERRUPTS WHILE READING
13A9  A0 0070 R                             MOV     AL,TIMER_OFL
13AC  C6 06 0070 R 00                       MOV     TIMER_OFL,0       ; GET OVERFLOW, AND RESET THE FLAG
13B1  8B 0E 006E R                          MOV     CX,TIMER_HIGH
13B5  8B 16 006C R                          MOV     DX,TIMER_LOW
13B9  EB EA                                 JMP     T1                ; TOD_RETURN
13BB  FA                            T3:     CLI                       ; NO INTERRUPTS WHILE WRITING
13BC  89 16 006C R                          MOV     TIMER_LOW,DX
13C0  89 0E 006E R                          MOV     TIMER_HIGH,CX     ; SET THE TIME
13C4  C6 06 0070 R 00                       MOV     TIMER_OFL,0       ; RESET OVERFLOW
13C9  EB DA                                 JMP     T1                ; TOD_RETURN
```

```
13CB  51                    T4A:    PUSH    CX                ;
13CC  B1 05                         MOV     CL,5              ;
13CE  D2 E0                         SAL     AL,CL             ; SHIFT PARM BITS LEFT 5 POSITIONS
13D0  86 C4                         XCHG    AL,AH             ; SAVE PARM
13D2  E4 61                         IN      AL,PORT_B         ; GET CURRENT PORT SETTINGS
13D4  24 9F                         AND     AL,10011111B      ; ISOLATE MUX BITS
13D6  0A C4                         OR      AL,AH             ; COMBINE PORT BITS/PARM BITS
13D8  E6 61                         OUT     PORT_B,AL         ; SET PORT TO NEW VALUE
13DA  59                            POP     CX
13DB  EB C8                         JMP     T1                ; TOD_RETURN
13DD                        TIME_OF_DAY     ENDP
                            ;---- INT 16 ---------------------------------------------------
                            ; KEYBOARD I/O
                            ;       THESE ROUTINES PROVIDE KEYBOARD SUPPORT
                            ; INPUT
                            ;       (AH)=0  READ THE NEXT ASCII CHARACTER STRUCK FROM THE
                            ;               KEYBOARD, RETURN THE RESULT IN (AL), SCAN CODE IN
                            ;               (AH)
                            ;       (AH)=1  SET THE Z FLAG TO INDICATE IF AN ASCII CHARACTER IS
                            ;               AVAILABLE TO BE READ.
                            ;               (ZF)=1 -- NO CODE AVAILABLE
                            ;               (ZF)=0 -- CODE IS AVAILABLE
                            ;               IF ZF = 0, THE NEXT CHARACTER IN THE BUFFER TO BE
                            ;               READ IS IN AX, AND THE ENTRY REMAINS IN THE BUFFER
                            ;       (AH)=2  RETURN THE CURRENT SHIFT STATUS IN AL REGISTER
                            ;               THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
                            ;               THE EQUATES FOR KB_FLAG
                            ;       (AH)=3  SET TYPAMATIC RATES.  THE TYPAMATIC RATE CAN BE
                            ;               CHANGED USING THE FOLLOWING FUNCTIONS:
                            ;               (AL)=0      RETURN TO DEFAULT.  RESTORES ORIGINAL
                            ;                           STATE. I.E. TYPAMATIC ON, NORMAL INITIAL
                            ;                           DELAY, AND NORMAL TYPAMATIC RATE.
                            ;               (AL)=1      INCREASE INITIAL DELAY.  THIS IS THE
                            ;                           DELAY BETWEEN THE FIRST CHARACTER AND
                            ;                           THE BURST OF TYPAMATIC CHARS.
                            ;               (AL)=2      HALF_RATE.  SLOWS TYPAMATIC CHARACTERS
                            ;                           BY ONE HALF.
                            ;               (AL)=3      COMBINES AL=1 AND AL=2.  INCREASES
                            ;                           INITIAL DELAY AND SLOWS TYPAMATIC
                            ;                           CHARACTERS BY ONE HALF.
                            ;               (AL)=4      TURN OFF TYPAMATIC CHARACTERS.  ONLY THE
                            ;                           FIRST CHARACTER IS HONORED.  ALL OTHERS
                            ;                           ARE IGNORED.
                            ;               AL IS RANGE CHECKED.  IF AL<0 OR AL>4 THE STATE
                            ;               REMAINS THE SAME.
                            ;               ***NOTE***  EACH TIME THE TYPAMATIC RATES ARE
                            ;               CHANGED ALL PREVIOUS STATES ARE REMOVED.  I.E. IF
                            ;               THE KEYBOARD IS IN THE HALF RATE MODE AND YOU WANT
                            ;               TO ADD AN INCREASE IN TYPAMATIC DELAY, YOU MUST
                            ;               CALL THIS ROUTINE WITH AH=3 AND AL=3.
                            ;       (AH)=4  ADJUST KEYBOARD BY THE VALUE IN AL AS FOLLOWS:
                            ;               (AL)=0       TURN OFF KEYBOARD CLICK.
                            ;               (AL)=1       TURN ON KEYBOARD CLICK.
                            ;               AL IS RANGE CHECKED.  THE STATE IS UNALTERED IF
                            ;               AL <> 1,0.
                            ; OUTPUT
                            ;       AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED
                            ;       ALL REGISTERS RETAINED
                            ;--------------------------------------------------------------
13DD                        KEYBOARD_IO     PROC    FAR
                            ASSUME  CS:CODE,DS:DATA
13DD  FB                            STI                       ; INTERRUPTS BACK ON
13DE  1E                            PUSH    DS                ; SAVE CURRENT DS
13DF  53                            PUSH    BX                ; SAVE BX TEMPORARILY
13E0  E8 138B R                     CALL    DDS               ; POINT DS AT BIOS DATA SEGMENT
13E3  0A E4                         OR      AH,AH             ; AH=0
13E5  74 0A                         JZ      K1                ; ASCII_READ
13E7  FE CC                         DEC     AH                ; AH=1
13E9  74 1E                         JZ      K2                ; ASCII_STATUS
13EB  FE CC                         DEC     AH                ; AH=2
13ED  74 2B                         JZ      K3                ; SHIFT_STATUS
13EF  EB 2E                         JMP     SHORT   K3_1
                            ;------ READ THE KEY TO FIGURE OUT WHAT TO DO
13F1                        K1:                               ; ASCII READ
13F1  FB                            STI                       ; INTERRUPTS BACK ON DURING LOOP
13F2  90                            NOP                       ; ALLOW AN INTERRUPT TO OCCUR
13F3  FA                            CLI                       ; INTERRUPTS BACK OFF
13F4  8B 1E 001A R                  MOV     BX,BUFFER_HEAD    ; GET POINTER TO HEAD OF BUFFER
13F8  3B 1E 001C R                  CMP     BX,BUFFER_TAIL    ; TEST END OF BUFFER
13FC  74 F3                         JZ      K1                ; LOOP UNTIL SOMETHING IN BUFFER
13FE  8B 07                         MOV     AX,[BX]           ; GET SCAN CODE AND ASCII CODE
1400  E8 144F R                     CALL    K4                ; MOVE POINTER TO NEXT POSITION
1403  89 1E 001A R                  MOV     BUFFER_HEAD,BX    ; STORE VALUE IN VARIABLE
1407  EB 43                         JMP     SHORT   RET_INT16
                            ;------ ASCII STATUS
1409  FA                    K2:     CLI                       ; INTERRUPTS OFF
140A  8B 1E 001A R                  MOV     BX,BUFFER_HEAD    ; GET HEAD POINTER
140E  3B 1E 001C R                  CMP     BX,BUFFER_TAIL    ; IF EQUAL (Z=1) THEN NOTHING THERE
1412  8B 07                         MOV     AX,[BX]
1414  FB                            STI                       ; INTERRUPTS BACK ON
1415  5B                            POP     BX                ; RECOVER REGISTER
1416  1F                            POP     DS                ; RECOVER SEGMENT
1417  CA 0002                       RET     2                 ; THROW AWAY FLAGS
                            ;------ SHIFT STATUS
141A  A0 0017 R             K3:     MOV     AL,KB_FLAG        ; GET THE SHIFT STATUS FLAGS
141D  EB 2D                         JMP     SHORT   RET_INT16
```

```
                               ;------- ADJUST KEY CLICK
141F  FE CC                    K3_1:   DEC     AH
1421  74 1A                            JZ      K3_3            ; AH=3, ADJUST TYPAMATIC
1423  FE CC                            DEC     AH              ; RANGE CHECK FOR AH=4
1425  75 25                            JNZ     RET_INT16       ; ILLEGAL FUNCTION CALL
1427  0A C0                            OR      AL,AL           ; TURN OFF KEYBOARD CLICK?
1429  75 07                            JNZ     K3_2            ; JUMP FOR RANGE CHECK
142B  80 26 0018 R FB                  AND     KB_FLAG_1,AND_MASK-CLICK_ON ; TURN OFF CLICK
1430  EB 1A                            JMP     SHORT RET_INT16
1432  3C 01                    K3_2:   CMP     AL,1            ; RANGE CHECK
1434  75 16                            JNE     RET_INT16       ; NOT IN RANGE, RETURN
1436  80 0E 0018 R 04                  OR      KB_FLAG_1,CLICK_ON ; TURN ON KEYBOARD CLICK
143B  EB 0F                            JMP     SHORT   RET_INT16
                               ;------ SET TYPAMATIC
143D  3C 04                    K3_3:   CMP     AL,4            ; CHECK FOR CORRECT RANGE
143F  7F 0B                            JG      RET_INT16       ; IF ILLEGAL VALUE IN AL IGNORE
1441  80 26 0088 R F1                  AND     KB_FLAG_2,0F1H  ; MASK OFF ANY OLD TYPAMATIC STATES
1446  D0 E0                            SHL     AL,1            ; SHIFT TO PROPER POSITION
1448  08 06 0088 R                     OR      KB_FLAG_2,AL
144C                           RET_INT16:
144C  5B                               POP     BX              ; RECOVER REGISTER
144D  1F                               POP     DS              ; RECOVER REGISTER
144E  CF                               IRET                    ; RETURN TO CALLER
144F                           KEYBOARD_IO     ENDP
                               ;------ INCREMENT A BUFFER POINTER
144F                           K4      PROC    NEAR
144F  43                               INC     BX              ; MOVE TO NEXT WORD IN LIST
1450  43                               INC     BX
1451  3B 1E 0082 R                     CMP     BX,BUFFER_END   ; AT END OF BUFFER?
1455  75 04                            JNE     K5              ; NO, CONTINUE
1457  8B 1E 0080 R                     MOV     BX,BUFFER_START ; YES, RESET TO BUFFER BEGINNING
145B  C3                       K5:     RET
145C                           K4      ENDP
                               ;------ TABLE OF SHIFT KEYS AND MASK VALUES
145C                           K6      LABEL   BYTE
145C  52                               DB      INS_KEY         ; INSERT KEY
145D  3A 45 46 38 1D                   DB      CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
1462  2A 36                            DB      LEFT_KEY,RIGHT_KEY
= 0008                         K6L     EQU     $-K6
                               ;------ SHIFT_MASK_TABLE
1464                           K7      LABEL   BYTE
1464  80                               DB      INS_SHIFT       ; INSERT MODE SHIFT
1465  40 20 10 08 04                   DB   CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
146A  02 01                            DB      LEFT_SHIFT,RIGHT_SHIFT
                               ;------ SCAN CODE TABLES
146C  1B FF 00 FF FF FF        K8              DB      27,-1,0,-1,-1,-1,30,-1
      1E FF
1474  FF FF FF 1F FF 7F                        DB      -1,-1,-1,31,-1,127,-1,17
      FF 11
147C  17 05 12 14 19 15                        DB      23,5,18,20,25,21,9,15
      09 0F
1484  10 1B 1D 0A FF 01                        DB      16,27,29,10,-1,1,19
      13
148B  04 06 07 08 0A 0B                        DB      4,6,7,8,10,11,12,-1,-1
      0C FF FF
1494  FF FF 1C 1A 18 03                        DB      -1,-1,28,26,24,3,22,2
      16 02
149C  0E 0D FF FF FF FF                        DB      14,13,-1,-1,-1,-1,-1,-1
      FF FF
14A4  20 FF                                    DB      ' ',-1
                               ;-------- CTL TABLE SCAN
14A6                           K9      LABEL   BYTE
14A6  5E 5F 60 61 62 63                        DB      94,95,96,97,98,99,100,101
      64 65
14AE  66 67 FF FF 77 FF                        DB      102,103,-1,-1,119,-1,132,-1
      84 FF
14B6  73 FF 74 FF 75 FF                        DB      115,-1,116,-1,117,-1,118,-1
      76 FF
14BE  FF                                       DB      -1
                               ;------- LC TABLE
14BF                           K10     LABEL   BYTE
14BF  1B 31 32 33 34 35                        DB      01BH,'1234567890-=',08H,09H
      36 37 38 39 30 2D
      3D 08 09
14CE  71 77 65 72 74 79                        DB      'qwertyuiop[]',0DH,-1,'asdfghjkl;',027H
      75 69 6F 70 5B 5D
      0D FF 61 73 64 66
      67 68 6A 6B 6C 3B
      27
14E7  60 FF 5C 7A 78 63                        DB      60H,-1,5CH,'zxcvbnm,./',-1,'*',-1,' '
      76 62 6E 6D 2C 2E
      2F FF 2A FF 20
14F8  FF                                       DB      -1
                               ;------ UC TABLE
14F9                           K11     LABEL   BYTE
14F9  1B 21 40 23 24 25                        DB      27,'!@#$',37,05EH,'&*()_+',08H,0
      5E 26 2A 28 29 5F
      2B 08 00
1508  51 57 45 52 54 59                        DB      'QWERTYUIOP{}',0DH,-1,'ASDFGHJKL:"'
      55 49 4F 50 7B 7D
      0D FF 41 53 44 46
      47 48 4A 4B 4C 3A
      22
1521  7E FF 7C 5A 58 43                        DB      07EH,-1,'|ZXCVBNM<>?',-1,0,-1,' ',-1
      56 42 4E 4D 3C 3E
      3F FF 00 FF 20 FF
```

```
                                        ;------ UC TABLE SCAN
1533                                    K12     LABEL   BYTE
1533  54 55 56 57 58 59                                 DB      84,85,86,87,88,89,90
      5A
153A  5B 5C 5D                                          DB      91,92,93
                                        ;------ ALT TABLE SCAN
153D                                    K13     LABEL   BYTE
153D  68 69 6A 6B 6C                                    DB      104,105,106,107,108
1542  6D 6E 6F 70 71                                    DB      109,110,111,112,113
                                        ;------ NUM STATE TABLE
1547                                    K14     LABEL   BYTE
1547  37 38 39 2D 34 35                                 DB      '789-456+1230.'
      36 2B 31 32 33 30
      2E
                                        ;------ BASE CASE TABLE
1554                                    K15     LABEL   BYTE
1554  47 48 49 FF 4B FF                                 DB      71,72,73,-1,75,-1,77
      4D
155B  FF 4F 50 51 52 53                                 DB      -1,79,80,81,82,83
                                        ;------ KEYBOARD INTERRUPT ROUTINE
1561                                    KB_INT  PROC    FAR
1561  FB                                        STI                     ; ALLOW FURTHER INTERRUPTS
1562  50                                        PUSH    AX
1563  53                                        PUSH    BX
1564  51                                        PUSH    CX
1565  52                                        PUSH    DX
1566  56                                        PUSH    SI
1567  57                                        PUSH    DI
1568  1E                                        PUSH    DS
1569  06                                        PUSH    ES
156A  FC                                        CLD                     ; FORWARD DIRECTION
156B  E8 138B R                                 CALL    DDS
156E  8A E0                                     MOV     AH,AL           ; SAVE SCAN CODE IN AH
                                        ;------ TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
1570  3C FF                                     CMP     AL,0FFH         ; IS THIS AN OVERRUN CHAR?
1572  75 1B                                     JNZ     K16             ; NO, TEST FOR SHIFT KEY
1574  BB 0080                                   MOV     BX,80H          ; DURATION OF ERROR BEEP
1577  B9 0048                                   MOV     CX,48H          ; FREQUENCY OF TONE
157A  E8 E035 R                                 CALL    KB_NOISE        ; BUFFER FULL BEEP
157D  80 26 0017 R F0                           AND     KB_FLAG,0F0H    ; CLEAR ALT,CLRL,LEFT AND RIGHT
                                                                        ; SHIFTS
1582  80 26 0018 R 0F                           AND     KB_FLAG_1,0FH   ; CLEAR POTENTIAL BREAK OF INS,CAPS
                                                                        ; ,NUM AND SCROLL SHIFT
1587  80 26 0088 R 1F                           AND     KB_FLAG_2,1FH   ; CLEAR FUNCTION STATES
158C  E9 164A R                                 JMP     K26             ; END OF INTERRUPT
                                        ;------ TEST FOR SHIFT KEYS
158F                                    K16:                            ; TEST_SHIFT
158F  24 7F                                     AND     AL,07FH         ; TURN OFF THE BREAK BIT
1591  0E                                        PUSH    CS
1592  07                                        POP     ES              ; ESTABLISH ADDRESS OF SHIFT TABLE
1593  BF 145C R                                 MOV     DI,OFFSET K6    ; SHIFT KEY TABLE
1596  B9 0008                                   MOV     CX,K6L          ; LENGTH
1599  F2/ AE                                    REPNE   SCASB           ; LOOK THROUGH THE TABLE FOR A
                                                                        ; MATCH
159B  8A C4                                     MOV     AL,AH           ; RECOVER SCAN CODE
159D  74 03                                     JE      K17             ; JUMP IF MATCH FOUND
159F  E9 163A R                                 JMP     K25             ; IF NO MATCH, THEN SHIFT NOT FOUND
                                        ;------ SHIFT KEY FOUND
15A2  81 EF 145D R                      K17:    SUB     DI,OFFSET K6+1  ; ADJUST PTR TO SCAN CODE MATCH
15A6  2E: 8A A5 1464 R                          MOV     AH,CS:K7[DI]    ; GET MASK INTO AH
15AB  A8 80                                     TEST    AL,80H          ; TEST FOR BREAK KEY
15AD  75 51                                     JNZ     K23             ; BREAK_SHIFT_FOUND
                                        ;------ SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
15AF  80 FC 10                                  CMP     AH,SCROLL_SHIFT
15B2  73 07                                     JAE     K18             ; IF SCROLL SHIFT OR ABOVE, TOGGLE
                                                                        ; KEY
                                        ;------ PLAIN SHIFT KEY, SET SHIFT ON
15B4  08 26 0017 R                              OR      KB_FLAG,AH      ; TURN ON SHIFT BIT
15B8  E9 164A R                                 JMP     K26             ; INTERRUPT_RETURN
                                        ;------ TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
15BB                                    K18:                            ; SHIFT-TOGGLE
15BB  F6 06 0017 R 04                           TEST    KB_FLAG, CTL_SHIFT ; CHECK CTL SHIFT STATE
15C0  75 78                                     JNZ     K25             ; JUMP IF CTL STATE
15C2  3C 52                                     CMP     AL, INS_KEY     ; CHECK FOR INSERT KEY
15C4  75 22                                     JNZ     K22             ; JUMP IF NOT INSERT KEY
15C6  F6 06 0017 R 08                           TEST    KB_FLAG, ALT_SHIFT ; CHECK FOR ALTERNATE SHIFT
15CB  75 6D                                     JNZ     K25             ; JUMP IF ALTERNATE SHIFT
15CD  F6 06 0017 R 20                           TEST    KB_FLAG, NUM_STATE ; CHECK FOR BASE STATE
15D2  75 0D                                     JNZ     K21             ; JUMP IF NUM LOCK IS ON
15D4  F6 06 0017 R 03                           TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT       ;
15D9  74 0D                                     JZ      K22             ; JUMP IF BASE STATE
15DB                                    K20:                            ; NUMERIC ZERO, NOT INSERT KEY
15DB  B8 5230                                   MOV     AX, 5230H       ; PUT OUT AN ASCII ZERO
15DE  E9 17EC R                                 JMP     K57             ; BUFFER_FILL
15E1                                    K21:                            ; MIGHT BE NUMERIC
15E1  F6 06 0017 R 03                           TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT
15E6  74 F3                                     JZ      K20             ; JUMP NUMERIC, NOT INSERT
15E8                                    K22:                            ; SHIFT TOGGLE KEY HIT; PROCESS IT
15E8  84 26 0018 R                              TEST    AH,KB_FLAG_1    ; IS KEY ALREADY DEPRESSED
15EC  75 5C                                     JNZ     K26             ; JUMP IF KEY ALREADY DEPRESSED
15EE  08 26 0018 R                              OR      KB_FLAG_1,AH    ; INDICATE THAT THE KEY IS
                                                                        ; DEPRESSED
15F2  30 26 0017 R                              XOR     KB_FLAG,AH      ; TOGGLE THE SHIFT STATE
15F6  3C 52                                     CMP     AL,INS_KEY      ; TEST FOR 1ST MAKE OF INSERT KEY
15F8  75 50                                     JNE     K26             ; JUMP IF NOT INSERT KEY
15FA  B8 5200                                   MOV     AX,INS_KEY*256  ; SET SCAN CODE INTO AH, 0 INTO AL
15FD  E9 17EC R                                 JMP     K57             ; PUT INTO OUTPUT BUFFER
```

```
                                ;------ BREAK SHIFT FOUND
1600                            K23:                            ; BREAK-SHIFT-FOUND
1600  80 FC 10                          CMP     AH,SCROLL_SHIFT ; IS THIS A TOGGLE KEY
1603  73 1A                             JAE     K24             ; YES, HANDLE BREAK TOGGLE
1605  F6 D4                             NOT     AH              ; INVERT MASK
1607  20 26 0017 R                      AND     KB_FLAG,AH      ; TURN OFF SHIFT BIT
160B  3C B8                             CMP     AL,ALT_KEY+80H  ; IS THIS ALTERNATE SHIFT RELEASE
160D  75 3B                             JNE     K26             ; INTERRUPT_RETURN
                                ;------ ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
160F  A0 0019 R                         MOV     AL,ALT_INPUT
1612  32 E4                             XOR     AH,AH           ; SCAN CODE OF 0
1614  88 26 0019 R                      MOV     ALT_INPUT,AH    ; ZERO OUT THE FIELD
1618  0A C0                             OR      AL,AL           ; WAS THE INPUT=0?
161A  74 2E                             JE      K26             ; INTERRUPT_RETURN
161C  E9 17F5 R                         JMP     K58             ; IT WASN'T, SO PUT IN BUFFER
161F                            K24:                            ; BREAK-TOGGLE
161F  3C BA                             CMP     AL,CAPS_KEY+BREAK_BIT ; SPECIAL CASE OF TOGGLE KEY
1621  75 0F                             JNE     K24_1           ; JUMP AROUND POTENTIAL UPDATE
1623  F6 06 0018 R 02                   TEST    KB_FLAG_1,CLICK_SEQUENCE
1628  74 08                             JZ      K24_1           ; JUMP IF NOT SPECIAL CASE
162A  80 26 0018 R FD                   AND     KB_FLAG_1,AND_MASK-CLICK_SEQUENCE ; MASK OFF MAKE
                                                                ; OF CLICK
162F  EB 19 90                          JMP     K26             ; INTERRUPT IS OVER
                                ;------ BREAK OF NORMAL TOGGLE
1632  F6 D4                     K24_1:  NOT     AH              ; INVERT MASK
1634  20 26 0018 R                      AND     KB_FLAG_1,AH    ; INDICATE NO LONGER DEPRESSED
1638  EB 10                             JMP     SHORT K26       ; INTERRUPT_RETURN
                                ;------ TEST FOR HOLD STATE
163A                            K25:                            ; NO-SHIFT-FOUND
163A  3C 80                             CMP     AL,80H          ; TEST FOR BREAK KEY
163C  73 0C                             JAE     K26             ; NOTHING FOR BREAK CHARS FROM HERE
                                                                ; ON
163E  F6 06 0018 R 08                   TEST    KB_FLAG_1,HOLD_STATE ; ARE WE IN HOLD STATE?
1643  74 0E                             JZ      K28             ; BRANCH AROUND TEST IF NOT
1645  80 26 0018 R F7                   AND     KB_FLAG_1,NOT HOLD_STATE ; TURN OFF THE HOLD STATE
                                                                ; BIT
164A                            K26:                            ; INTERRUPT-RETURN
164A  07                                POP     ES
164B  1F                                POP     DS
164C  5F                                POP     DI
164D  5E                                POP     SI
164E  5A                                POP     DX
164F  59                                POP     CX
1650  5B                                POP     BX
1651  58                                POP     AX              ; RESTORE STATE
1652  CF                                IRET                    ; RETURN, INTERRUPTS BACK ON WITH
                                                                ; FLAG CHANGE
                                ;------ NOT IN  HOLD STATE, TEST FOR SPECIAL CHARS
1653                            K28:                            ; NO-HOLD-STATE
1653  F6 06 0017 R 08                   TEST    KB_FLAG,ALT_SHIFT ; ARE WE IN ALTERNATE SHIFT
1658  75 03                             JNZ     K29             ; JUMP IF ALTERNATE SHIFT
165A  E9 1749 R                         JMP     K38             ; JUMP IF NOT ALTERNATE
                                ;------ TEST FOR ALT+CTRL KEY SEQUENCES
165D                            K29:                            ; TEST-RESET
165D  F6 06 0017 R 04                   TEST    KB_FLAG,CTL_SHIFT ; ARE WE IN CONTROL SHIFT ALSO
1662  74 69                             JZ      K31             ; NO_RESET
1664  3C 53                             CMP     AL,DEL_KEY      ; SHIFT STATE IS THERE, TEST KEY
1666  75 09                             JNE     K29_1           ; NO_RESET
                                ;------ CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
1668  C7 06 0072 R 1234                 MOV     RESET_FLAG, 1234H ; SET FLAG FOR RESET FUNCTION
166E  E9 0043 R                         JMP     NEAR PTR RESET  ; JUMP TO POWER ON DIAGNOSTICS
1671  3C 52                     K29_1:  CMP     AL,INS_KEY      ; CHECK FOR RESET WITH DIAGNOSTICS
1673  75 09                             JNE     K29_2           ; CHECK FOR OTHER
                                                                ; ALT-CTRL-SEQUENCES
                                ;------ ALT-CTRL-INS HAS BEEN FOUND
1675  C7 06 0072 R 4321                 MOV     RESET_FLAG,4321H ; SET FLAG FOR DIAGNOSTICS
167B  E9 0043 R                         JMP     NEAR PTR RESET  ; LEVEL 1 DIAGNOSTICS
167E  3C 3A                     K29_2:  CMP     AL,CAPS_KEY     ; CHECK FOR KEYBORAD CLICK TOGGLE
1680  75 13                             JNE     K29_3           ; CHECK FOR SCREEN ADJUSTMENT
                                ;------ ALT+CTRL+CAPSLOCK HAS BEEN FOUND
1682  F6 06 0018 R 02                   TEST    KB_FLAG_1,CLICK_SEQUENCE
1687  75 C1                             JNZ     K26             ; JUMP IF SEQUENCE HAS ALREADY
                                                                ; OCCURED
1689  80 36 0018 R 04                   XOR     KB_FLAG_1,CLICK_ON ; TOGGLE BIT FOR AUDIO KEYSTROKE
                                                                ; FEEDBACK
168E  80 0E 0018 R 02                   OR      KB_FLAG_1,CLICK_SEQUENCE ; SET CLICK_SEQUENCE STATE
1693  EB B5                             JMP     SHORT K26       ; INTERRUPT IS OVER
1695  3C 4D                     K29_3:  CMP     AL,RIGHT_ARROW  ; ADJUST SCREEN TO THE RIGHT?
1697  75 12                             JNE     K29_4           ; LOOK FOR RIGHT ADJUSTMENT
1699  E8 186E R                         CALL    GET_POS         ; GET THE # OF POSITIONS SCREEN IS
                                                                ; SHIFTED
169C  3C FC                             CMP     AL,0-RANGE      ; IS SCREEN SHIFTED AS FAR AS
                                                                ; POSSIBLE?
169E  7C AA                             JL      K26             ; OUT OF RANGE
16A0  FE 0E 0089 R                      DEC     HORZ_POS        ; SHIFT VALUE TO THE RIGHT
16A4  FE C8                             DEC     AL              ; DECREASE RANGE VALUE
16A6  E8 187A R                         CALL    PUT_POS         ; RESTORE STORAGE LOCATION
16A9  EB 14                             JMP     SHORT K29_5     ; ADJUST
16AB  3C 4B                     K29_4:  CMP     AL,LEFT_ARROW   ; ADJUST SREEN TO THE LEFT?
16AD  75 1E                             JNE     K31             ; NOT AN ALT_CTRL SEQUENCE
16AF  E8 186E R                         CALL    GET_POS         ; GET NUMBER OF POSITIONS SCREEN IS
                                                                ; SHIFTED
16B2  3C 04                             CMP     AL,RANGE        ; IS SCREEN SHIFTED AS FAR AS
                                                                ; POSSIBLE?
16B4  7F 94                             JG      K26
16B6  FE 06 0089 R                      INC     HORZ_POS        ; SHIFT SCREEN TO THE LEFT
16BA  FE C0                             INC     AL              ; INCREASE NUMBER OF POSITIONS
                                                                ; SCREEN IS SHIFTED
16BC  E8 187A R                         CALL    PUT_POS         ; PUT POSTION BACK IN STORAGE
```

```
16BF  B0 02                    K29_5:  MOV     AL,2              ; ADJUST
16C1  BA 03D4                          MOV     DX,3D4H           ; ADDRESS TO CRT CONTROLLER
16C4  EE                               OUT     DX,AL
16C5  A0 0089 R                        MOV     AL,HORZ_POS       ; COLUMN POSITION
16C8  42                               INC     DX                ; POINT AT DATA REGISTER
16C9  EE                               OUT     DX,AL             ; MOV POSITION
16CA  E9 164A R                        JMP     K26
                               ;------ IN ALTERNATE SHIFT, RESET NOT FOUND
16CD                           K31:                              ; NO-RESET
16CD  3C 39                            CMP     AL,57             ; TEST FOR SPACE KEY
16CF  75 29                            JNE     K32               ; NOT THERE
16D1  B0 20                            MOV     AL,' '            ; SET SPACE CHAR
16D3  E9 17EC R                        JMP     K57               ; BUFFER_FILL
                               ;----- ALT-INPUT-TABLE
16D6                           K30     LABEL   BYTE
16D6  52 4F 50 51 4B 4C                DB      82,79,80,81,75,76,77
      4D
16DD  47 48 49                         DB      71,72,73          ; 10 NUMBERS ON KEYPAD
                               ;------ SUPER-SHIFT-TABLE
16E0  10 11 12 13 14 15                DB      16,17,18,19,20,21,22,23 ; A-Z TYPEWRITER CHARS
      16 17
16E8  18 19 1E 1F 20 21                DB      24,25,30,31,32,33,34,35
      22 23
16F0  24 25 26 2C 2D 2E                DB      36,37,38,44,45,46,47,48
      2F 30
16F8  31 32                            DB      49,50
                               ;------ LOOK FOR KEY PAD ENTRY
16FA                           K32:                              ; ALT-KEY-PAD
16FA  BF 16D6 R                        MOV     DI,OFFSET K30     ; ALT-INPUT-TABLE
16FD  B9 000A                          MOV     CX,10             ; LOOK FOR ENTRY USING KEYPAD
1700  F2/ AE                           REPNE   SCASB             ; LOOK FOR MATCH
1702  75 13                            JNE     K33               ; NO_ALT_KEYPAD
1704  81 EF 16D7 R                     SUB     DI,OFFSET K30+1   ; DI NOW HAS ENTRY VALUE
1708  A0 0019 R                        MOV     AL,ALT_INPUT      ; GET THE CURRENT BYTE
170B  B4 0A                            MOV     AH,10             ; MULTIPLY BY 10
170D  F6 E4                            MUL     AH
170F  03 C7                            ADD     AX,DI             ; ADD IN THE LATEST ENTRY
1711  A2 0019 R                        MOV     ALT_INPUT,AL      ; STORE IT AWAY
1714  E9 164A R                        JMP     K26               ; THROW AWAY THAT KEYSTROKE
                               ;------ LOOK FOR SUPERSHIFT ENTRY
1717                           K33:                              ; NO-ALT-KEYPAD
1717  C6 06 0019 R 00                  MOV     ALT_INPUT,0       ; ZERO ANY PREVIOUS ENTRY INTO
                                                                 ; INPUT
171C  B9 001A                          MOV     CX,26             ; DI,ES ALREADY POINTING
171F  F2/ AE                           REPNE   SCASB             ; LOOK FOR MATCH IN ALPHABET
1721  75 05                            JNE     K34               ; NOT FOUND, FUNCTION KEY OR OTHER
1723  32 C0                            XOR     AL,AL             ; ASCII CODE OF ZERO
1725  E9 17EC R                        JMP     K57               ; PUT IT IN THE BUFFER
                               ;------ LOOK FOR TOP ROW OF ALTERNATE SHIFT
1728                           K34:                              ; ALT-TOP-ROW
1728  3C 02                            CMP     AL,2              ; KEY WITH '1' ON IT
172A  72 0C                            JB      K35               ;  NOT ONE OF INTERESTING KEYS
172C  3C 0E                            CMP     AL,14             ;  IS IT IN THE REGION?
172E  73 08                            JAE     K35               ;  ALT-FUNCTION
1730  80 C4 76                         ADD     AH,118            ;  CONVERT PSUEDO SCAN CODE TO
                                                                 ; RANGE
1733  32 C0                            XOR     AL,AL             ;  INDICATE AS SUCH
1735  E9 17EC R                        JMP     K57               ;  BUFFER_FILL
                               ;------ TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
1738                           K35:                              ; ALT-FUNCTION
1738  3C 3B                            CMP     AL,59             ; TEST FOR IN TABLE
173A  73 03                            JAE     K37               ; ALT-CONTINUE
173C                           K36:                              ; CLOSE-RETURN
173C  E9 164A R                        JMP     K26               ; IGNORE THE KEY
173F                           K37:                              ; ALT-CONTINUE
173F  3C 47                            CMP     AL,71             ; IN KEYPAD REGION
1741  73 F9                            JAE     K36               ; IF SO, IGNORE
1743  BB 153D R                        MOV     BX,OFFSET K13     ; ALT SHIFT PSEUDO SCAN TABLE
1746  E9 1863 R                        JMP     K63               ; TRANSLATE THAT
                               ;------ NOT IN ALTERNATE SHIFT
1749                           K38:                              ; NOT-ALT-SHIFT
1749  F6 06 0017 R 04                  TEST    KB_FLAG,CTL_SHIFT ; ARE WE IN CONTROL SHIFT?
174E  74 34                            JZ      K44               ; NOT-CTL-SHIFT
                               ;------ CONTROL SHIFT, TEST SPECIAL CHARACTERS
                               ;------ TEST FOR BREAK AND PAUSE KEYS
1750  3C 46                            CMP     AL,SCROLL_KEY     ; TEST FOR BREAK
1752  75 19                            JNE     K41               ; NO-BREAK
1754  8B 1E 001A R                     MOV     BX,BUFFER_HEAD    ; GET CURRENT BUFFER HEAD
1758  C6 06 0071 R 80                  MOV     BIOS_BREAK,80H    ; TURN ON BIOS_BREAK BIT
175D  CD 1B                            INT     1BH               ; BREAK INTERRUPT VECTOR
175F  2B C0                            SUB     AX,AX             ; PUT OUT DUMMY CHARACTER
1761  89 07                            MOV     [BX],AX           ; PUT DUMMY CHAR AT BUFFER HEAD
1763  E8 144F R                        CALL    K4                ; UPDATE BUFFER POINTER
1766  89 1E 001C R                     MOV     BUFFER_TAIL,BX    ; UPDATE TAIL
176A  E9 164A R                        JMP     K26               ; DONE WITH INTERUPT
176D                           K41:                              ; NO-PAUSE
                               ;------ TEST SPECIAL CASE KEY 55
176D  3C 37                            CMP     AL,55
176F  75 06                            JNE     K42               ; NOT-KEY-55
1771  B8 7200                          MOV     AX,114*256        ; START/STOP PRINTING SWITCH
1774  EB 76 90                         JMP     K57               ; BUFFER_FILL
```

```
                                ;------ SET UP TO TRANSLATE CONTROL SHIFT
1777                            K42:                              ; NOT-KEY-55
1777  BB 146C R                         MOV     BX,OFFSET K8      ; SET UP TO TRANSLATE CTL
177A  3C 3B                             CMP     AL,59             ; IS IT IN TABLE?
177C  72 6A                             JB      K56               ; YES, GO TRANSLATE CHAR
                                                                  ; CTL-TABLE-TRANSLATE
177E  BB 14A6 R                         MOV     BX,OFFSET K9      ; CTL TABLE SCAN
1781  E9 1863 R                         JMP     K63               ; TRANSLATE_SCAN
                                ;------ NOT IN CONTROL SHIFT
1784                            K44:                              ; NOT-CTL-SHIFT
1784  3C 47                             CMP     AL,71             ; TEST FOR KEYPAD REGION
1786  73 1F                             JAE     K48               ; HANDLE KEYPAD REGION
1788  F6 06 0017 R 03                   TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
178D  74 4E                             JZ      K54               ; TEST FOR SHIFT STATE
                                ;------ UPPER CASE, HANDLE SPECIAL CASES
178F  3C 0F                             CMP     AL,15             ; BACK TAB KEY
1791  75 05                             JNE     K46               ; NOT-BACK-TAB
1793  B8 0F00                           MOV     AX,15*256         ; SET PSEUDO SCAN CODE
1796  EB 54                             JMP     SHORT K57         ; BUFFER_FILL
1798                            K46:                              ; NOT-PRINT-SCREEN
1798  3C 3B                             CMP     AL,59             ; FUNCTION KEYS
179A  72 06                             JB      K47               ; NOT-UPPER-FUNCTION
179C  BB 1533 R                         MOV     BX,OFFSET K12     ; UPPER CASE PSEUDO SCAN CODES
179F  E9 1863 R                         JMP     K63               ; TRANSLATE_SCAN
17A2                            K47:                              ; NOT-UPPER-FUNCTION
17A2  BB 14F9 R                         MOV     BX,OFFSET K11     ; POINT TO UPPER CASE TABLE
17A5  EB 41                             JMP     SHORT K56         ; OK, TRANSLATE THE CHAR
                                ;------ KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
17A7                            K48:                              ; KEYPAD-REGION
17A7  F6 06 0017 R 20                   TEST    KB_FLAG,NUM_STATE ; ARE WE IN NUM_LOCK?
17AC  75 21                             JNZ     K52               ; TEST FOR SURE
17AE  F6 06 0017 R 03                   TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT
                                                                  ; STATE
17B3  75 21                             JNZ     K53               ; IF SHIFTED, REALLY NUM STATE
                                ;------ BASE CASE FOR KEYPAD
17B5                            K49:                              ; BASE-CASE
17B5  3C 4A                             CMP     AL,74             ; SPECIAL CASE FOR A COUPLE OF KEYS
17B7  74 0C                             JE      K50               ; MINUS
17B9  3C 4E                             CMP     AL,78
17BB  74 0D                             JE      K51
17BD  2C 47                             SUB     AL,71             ; CONVERT ORIGIN
17BF  BB 1554 R                         MOV     BX,OFFSET K15     ; BASE CASE TABLE
17C2  E9 1865 R                         JMP     K64               ; CONVERT TO PSEUDO SCAN
17C5  B8 4A2D                   K50:    MOV     AX,74*256+'-'     ; MINUS
17C8  EB 22                             JMP     SHORT K57         ; BUFFER_FILL
17CA  B8 4E2B                   K51:    MOV     AX,78*256+'+'     ; PLUS
17CD  EB 1D                             JMP     SHORT K57         ; BUFFER_FILL
                                ;------ MIGHT BE NUM LOCK, TEST SHIFT STATUS
17CF                            K52:                              ; ALMOST-NUM-STATE
17CF  F6 06 0017 R 03                   TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
17D4  75 DF                             JNZ     K49               ; SHIFTED TEMP OUT OF NUM STATE
17D6                            K53:                              ; REALLY_NUM_STATE
17D6  2C 46                             SUB     AL,70             ; CONVERT ORIGIN
17D8  BB 1547 R                         MOV     BX,OFFSET K14     ; NUM STATE TABLE
17DB  EB 0B                             JMP     SHORT K56         ; TRANSLATE_CHAR
                                ;------ PLAIN OLD LOWER CASE
17DD                            K54:                              ; NOT-SHIFT
17DD  3C 3B                             CMP     AL,59             ; TEST FOR FUNCTION KEYS
17DF  72 04                             JB      K55               ; NOT-LOWER-FUNCTION
17E1  32 C0                             XOR     AL,AL             ; SCAN CODE IN AH ALREADY
17E3  EB 07                             JMP     SHORT K57         ; BUFFER_FILL
17E5                            K55:                              ; NOT-LOWER-FUNCTION
17E5  BB 14BF R                         MOV     BX,OFFSET K10     ; LC TABLE
                                ;------ TRANSLATE THE CHARACTER
17E8                            K56:                              ; TRANSLATE-CHAR
17E8  FE C8                             DEC     AL                ; CONVERT ORIGIN
17EA  2E: D7                            XLAT    CS:K11            ; CONVERT THE SCAN CODE TO ASCII
                                ;------ PUT CHARACTER INTO BUFFER
17EC                            K57:                              ; BUFFER-FILL
17EC  3C FF                             CMP     AL,-1             ; IS THIS AN IGNORE CHAR?
17EE  74 1F                             JE      K59               ; YES, DO NOTHING WITH IT
17F0  80 FC FF                          CMP     AH,-1             ; LOOK FOR -1 PSEUDO SCAN
17F3  74 1A                             JE      K59               ; NEAR_INTERRUPT_RETURN
                                ;------ HANDLE THE CAPS LOCK PROBLEM
17F5                            K58:                              ; BUFFER-FILL-NOTEST
17F5  F6 06 0017 R 40                   TEST    KB_FLAG,CAPS_STATE ; ARE WE IN CAPS LOCK STATE?
17FA  74 20                             JZ      K61               ; SKIP IF NOT
                                ;------ IN CAPS LOCK STATE
17FC  F6 06 0017 R 03                   TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT
                                                                  ; STATE
1801  74 0F                             JZ      K60               ; IF NOT SHIFT, CONVERT LOWER TO
                                                                  ; UPPER
                                ;------ CONVERT ANY UPPER CASE TO LOWER CASE
1803  3C 41                             CMP     AL,'A'            ; FIND OUT IF ALPHABETIC
1805  72 15                             JB      K61               ; NOT_CAPS_STATE
1807  3C 5A                             CMP     AL,'Z'
1809  77 11                             JA      K61               ; NOT_CAPS_STATE
180B  04 20                             ADD     AL,'a'-'A'        ; CONVERT TO LOWER CASE
180D  EB 0D                             JMP     SHORT K61         ; NOT_CAPS_STATE
180F                            K59:                              ; NEAR-INTERRUPT-RETURN
180F  E9 164A R                         JMP     K26               ; INTERRUPT_RETURN
                                ;------ CONVERT ANY LOWER CASE TO UPPER CASE
1812                            K60:                              ; LOWER-TO-UPPER
1812  3C 61                             CMP     AL,'a'            ; FIND OUT IF ALPHABETIC
1814  72 06                             JB      K61               ; NOT_CAPS_STATE
1816  3C 7A                             CMP     AL,'z'
1818  77 02                             JA      K61               ; NOT_CAPS_STATE
181A  2C 20                             SUB     AL,'a'-'A'        ; CONVERT TO UPPER CASE
```

```
181C                          K61:                              ; NOT-CAPS-STATE
181C  8B 1E 001C R                    MOV     BX,BUFFER_TAIL    ; GET THE END POINTER TO THE BUFFER
1820  8B F3                           MOV     SI,BX             ; SAVE THE VALUE
1822  E8 144F R                       CALL    K4                ; ADVANCE THE TAIL
1825  3B 1E 001A R                    CMP     BX,BUFFER_HEAD    ; HAS THE BUFFER WRAPPED AROUND?
1829  75 1D                           JNE     K61_1             ; BUFFER_FULL_BEEP
182B  53                              PUSH    BX                ; SAVE BUFFER_TAIL
182C  BB 0080                         MOV     BX,080H           ; DURATION OF ERROR BEEP
182F  B9 0048                         MOV     CX,48H            ; FREQUENCY OF ERROR BEEP HALF TONE
1832  E8 E035 R                       CALL    KB_NOISE          ; OUTPUT NOISE
1835  80 26 0017 R F0                 AND     KB_FLAG,0F0H      ; CLEAR ALT,CLRL,LEFT AND RIGHT
                                                                ; SHIFTS
183A  80 26 0018 R 0F                 AND     KB_FLAG_1,0FH     ; CLEAR POTENTIAL BREAK OF INS,CAPS
                                                                ; ,NUM AND SCROLL SHIFT
183F  80 26 0088 R 1F                 AND     KB_FLAG_2,1FH     ; CLEAR FUNCTION STATES
1844  5B                              POP     BX                ; RETRIEVE BUFFER TAIL
1845  E9 164A R                       JMP     K26               ; RETURN FROM INTERRUPT
1848  F6 06 0018 R 04         K61_1:  TEST    KB_FLAG_1,CLICK_ON ; IS AUDIO FEEDBACK ENABLED?
184D  74 0B                           JZ      K61_2             ; NO, JUST PUT IN BUFFER
184F  53                              PUSH    BX                ; SAVE BUFFER_TAIL VALUE
1850  BB 0001                         MOV     BX,1H             ; DURATION OF CLICK
1853  B9 0010                         MOV     CX,10H            ; FREQUENCY OF CLICK
1856  E8 E035 R                       CALL    KB_NOISE          ; OUTPUT AUDIO FEEDBACK OF KEY
                                                                ; STROKE
1859  5B                              POP     BX                ; RETRIEVE BUFFER_TAIL VALUE
185A  89 04                   K61_2:  MOV     [SI],AX           ; STORE THE VALUE
185C  89 1E 001C R                    MOV     BUFFER_TAIL,BX    ; MOVE THE POINTER UP
1860  E9 164A R                       JMP     K26               ; INTERRUPT_RETURN
                              ;------ TRANSLATE SCAN FOR PSEUDO SCAN CODES
1863                          K63:                              ; TRANSLATE-SCAN
1863  2C 3B                           SUB     AL,59             ; CONVERT ORIGIN TO FUNCTION KEYS
1865                          K64:                              ; TRANSLATE-SCAN-ORGD
1865  2E: D7                          XLAT    CS:K9             ; CTL TABLE SCAN
1867  8A E0                           MOV     AH,AL             ; PUT VALUE INTO AH
1869  32 C0                           XOR     AL,AL             ; ZERO ASCII CODE
186B  E9 17EC R                       JMP     K57               ; PUT IT INTO THE BUFFER
186E                          KB_INT  ENDP
                              ;-------------------------------------------------------------------
                              ;GET_POS
                              ;       THIS ROUTINE WILL SHIFT THE VALUE STORED IN THE HIGH NIBBLE
                              ;       OF THE VARIABLE VAR_DELAY TO THE LOW NIBBLE.
                              ;INPUT
                              ;       NONE.  IT IS ASSUMED THAT DS POINTS AT THE BIOS DATA AREA
                              ;OUTPUT
                              ;       AL CONTAINS THE SHIFTED VALUE.
                              ;-------------------------------------------------------------------
186E                          GET_POS PROC    NEAR
186E  51                              PUSH    CX                ; SAVE SHIFT REGISTER
186F  A0 0086 R                       MOV     AL,BYTE PTR VAR_DELAY ; GET STORAGE LOCATION
1872  24 F0                           AND     AL,0F0H           ; MASK OFF LOW NIBBLE
1874  B1 04                           MOV     CL,4              ; SHIFT OF FOUR BIT POSITIONS
1876  D2 F8                           SAR     AL,CL             ; SHIFT THE VALUE SIGN EXTENDED
1878  59                              POP     CX                ; RESTORE THE VALUE
1879  C3                              RET
187A                          GET_POS ENDP
                              ;-------------------------------------------------------------------
                              ;PUT_POS
                              ;       THIS ROUTINE WILL TAKE THE VALUE IN LOW ORDER NIBBLE IN
                              ;       AL AND STORE IT IN THE HIGH ORDER OF VAR_DELAY
                              ;INPUT
                              ;       AL CONTAINS THE VALUE FOR STORAGE
                              ;OUTPUT
                              ;       NONE.
                              ;-------------------------------------------------------------------
187A                          PUT_POS PROC    NEAR
187A  51                              PUSH    CX                ; SAVE REGISTER
187B  B1 04                           MOV     CL,4              ; SHIFT COUNT
187D  D2 E0                           SHL     AL,CL             ; PUT IN HIGH ORDER NIBBLE
187F  8A 0E 0086 R                    MOV     CL,BYTE PTR VAR_DELAY ; GET DATA BYTE
1883  80 E1 0F                        AND     CL,0FH            ; CLEAR OLD VALUE IN HIGH NIBBLE
1886  0A C1                           OR      AL,CL             ; COMBINE HIGH AND LOW NIBBLES
1888  A2 0086 R                       MOV     BYTE PTR VAR_DELAY,AL ; PUT IN POSITION
188B  59                              POP     CX                ; RESTORE REGISTER
188C  C3                              RET
188D                          PUT_POS ENDP
                              ;-------------------------------------------------------------------
                              ; MANUFACTURING ACTIVITY SIGNAL ROUTINE - INVOKED THROUGH THE TIMER
                              ; TICK ROUTINE DURING MANUFACTRUING ACTIVITIES . (ACCESSED THROUGH
                              ; INT 1CH)
                              ;-------------------------------------------------------------------
188D                          MFG_TICK        PROC    FAR
188D  50                              PUSH    AX
188E  2B C0                           SUB     AX,AX             ; SEND A 00 TO PORT 13 AS A
                                                                ; ACTIVITY SIGNAL
1890  E6 13                           OUT     13H,AL
1892  E4 61                           IN      AL,PORT_B         ; FLIP SPEAKER DATA TO OPPOSITE
                                                                ; SENSE
1894  8A E0                           MOV     AH,AL             ; SAVE ORIG SETTING
1896  80 E4 9D                        AND     AH,10011101B      ; MAKE SURE MUX IS -> RIGHT AND
                                                                ; ISOLATE SPEAKER BIT
1899  F6 D0                           NOT     AL                ; FLIP ALL BITS
189B  24 02                           AND     AL,00000010B      ; ISOLATE SPEAKER DATA BIT (NOW IN
                                                                ; OPPOSITE SENSE)
189D  0A C4                           OR      AL,AH             ; COMBINE WITH ORIG. DATA FROM
                                                                ; PORT B
189F  0C 10                           OR      AL,00010000B      ; AND DISABLE INTERNAL SPEAKER
18A1  E6 61                           OUT     PORT_B,AL
18A3  B0 20                           MOV     AL,20H            ; EOI TO INTR. CHIP
18A5  E6 20                           OUT     20H,AL
18A7  58                              POP     AX
18A8  CF                              IRET
18A9                          MFG_TICK        ENDP
```

```
                              ;------------------------------------------------------------
                              ;       CONVERT AND PRINT ASCII CODE
                              ;
                              ;       AL MUST CONTAIN NUMBER TO BE CONVERTED.
                              ;       AX AND BX DESTROYED.
                              ;------------------------------------------------------------
18A9                          XPC_BYTE        PROC    NEAR
18A9  50                              PUSH    AX              ; SAVE FOR LOW NIBBLE DISPLAY
18AA  B1 04                           MOV     CL,4            ; SHIFT COUNT
18AC  D2 E8                           SHR     AL,CL           ; NIBBLE SWAP
18AE  E8 18B4 R                       CALL    XLAT_PR         ; DO THE HIGH NIBBLE DISPLAY
18B1  58                              POP     AX              ; RECOVER THE NIBBLE
18B2  24 0F                           AND     AL,0FH          ; ISOLATE TO LOW NIBBLE
                                                              ; FALL INTO LOW NIBBLE CONVERSION
18B4                          XLAT_PR PROC    NEAR            ; CONVERT 00-0F TO ASCII CHARACTER
18B4  04 90                           ADD     AL,090H         ; ADD FIRST CONVERSION FACTOR
18B6  27                              DAA                     ; ADJUST FOR NUMERIC AND ALPHA
                                                              ; RANGE
18B7  14 40                           ADC     AL,040H         ; ADD CONVERSION AND ADJUST LOW
                                                              ; NIBBLE
18B9  27                              DAA                     ; ADJUST HIGH NIBBLE TO ASCII RANGE
18BA                          PRT_HEX PROC    NEAR
18BA  53                              PUSH    BX
18BB  B4 0E                           MOV     AH,14           ; DISPLAY CHARACTER IN AL
18BD  B7 00                           MOV     BH,0
18BF  CD 10                           INT     10H             ; CALL VIDEO_IO
18C1  5B                              POP     BX
18C2  C3                              RET
18C3                          PRT_HEX ENDP
18C3                          XLAT_PR ENDP
18C3                          XPC_BYTE        ENDP
                              ;CONTROL IS PASSED HERE WHEN THERE ARE NO PARALLEL PRINTERS
                              ;ATTACHED. CX HAS EQUIPMENT FLAG,DS POINTS AT DATA (40H)
                              ;DETERMINE WHICH RS232 CARD (0,1) TO USE
18C3                          REPRINT PROC NEAR
18C3  2B D2                   B1_A:   SUB     DX,DX           ;ASSUME TO USE CARD 0
18C5  F6 C5 04                        TEST    CH,00000100B    ;UNLESS THERE ARE TWO CARDS
18C8  74 01                           JE      B10_1           ;IN WHICH CASE,
18CA  42                              INC     DX              ;USE CARD 1
                              ;DETERMINE WHICH FUNCTION IS BEING CALLED
18CB  0A E4                   B10_1:  OR      AH,AH           ;TEST FOR AH = 0
18CD  74 41                           JZ      B12             ;GO PRINT CHAR
18CF  FE CC                           DEC     AH              ;TEST FOR AH = 1
18D1  74 1D                           JZ      B11             ;GO DO INIT
18D3  FE CC                           DEC     AH              ;TEST FOR AH = 2
18D5  75 16                           JNZ     SHORT B10_3     ;IF NOT VALID, RETURN
                                                              ;ELSE...
                              ;GET STATUS FROM RS232 PORT
18D7  50                              PUSH    AX              ;SAVE AL
18D8  B4 03                           MOV     AH,03H          ;USE THE GET COMMO PORT
18DA  CD 14                           INT     014H            ;STATUS FUNCTION OF INT14
18DC  E8 1925 R                       CALL    FAKE            ;FAKE WILL MAP ERROR BITS FROM
                                                              ;RS232 TO CORRESPONDING ONES
                                                              ;FOR THE PRINTER
18DF  58                              POP     AX              ;RESTORE AL
18E0  0A F6                           OR      DH,DH           ;CHECK IF ANY FLAGS WERE SET
18E2  74 07                           JZ      B10_2
18E4  8A E6                           MOV     AH,DH           ;MOVE FAKED ERROR CONDITION TO AH
18E6  80 E4 FE                        AND     AH,0FEH
18E9  EB 02                           JMP     SHORT B10_3     ;THEN RETURN
18EB  B4 90                   B10_2:  MOV     AH,090H         ;MOVE IN STATUS FOR 'CORRECT'
                                                              ; RETURN
18ED  E9 F00D R               B10_3:  JMP     B1
                              ;INIT COMMO PORT    --- DX HAS WHICH CARD TO INIT.
                              ;MOVE TIME OUT VALUE FROM PRINTER TO RS232 TIME OUT VALUE
18F0  8B F2                   B11:    MOV     SI,DX           ;SI GETS OFFSET INTO THE TABLE
18F2  A0 0078 R                       MOV     AL,PRINT_TIM_OUT
18F5  04 0A                           ADD     AL,0AH          ; INCREASE DELAY
18F7  88 84 007C R                    MOV     RS232_TIM_OUT[SI],AL
18FB  50                              PUSH    AX              ;SAVE AL
18FC  B0 87                           MOV     AL,087H         ;SET INIT FOR: 1200 BAUD
                                                              ;              8 BIT WRD LNG
                                                              ;              NO PARITY
                                                              ;              2 STOP BITS
18FE  2A E4                           SUB     AH,AH           ;AH=0 IS COMMO INIT FUNCTION
1900  CD 14                           INT     014H            ;DO INIT
1902  E8 1925 R                       CALL    FAKE            ;FAKE WILL MAP ERROR BITS FROM
                                                              ;RS232 TO CORRESPONDING ONES
                                                              ;FOR THE PRINTER
1905  58                              POP     AX              ;RESTORE AL
1906  8A E6                           MOV     AH,DH           ;IF DH IS RETURNED ZERO, MEANING
1908  0A E4                           OR      AH,AH           ;NO ERRORS RETURN IT FOR THAT'S THE
                                                              ;'CORRECT' RETURN FROM AN ERROR
                                                              ; FREE INIT
190A  74 E1                           JE      B10_3
190C  B4 A8                           MOV     AH,0A8H
190E  EB DD                           JMP     SHORT B10_3     ;THEN RETURN
```

```
                                ;PRINT CHAR TO SERIAL PORT
                                ;DX = RS232 CARD TO BE USED: AL HAS CHAR TO BE PRINTED
1910  50                        B12:    PUSH    AX              ;SAVE AL
1911  B4 01                             MOV     AH,01           ;1 IS SEND A CHAR DOWN COMMO LINE
1913  CD 14                             INT     014H            ;SEND THE CHAR
1915  E8 1925 R                         CALL    FAKE            ;FAKE WILL MAP ERROR BITS FROM
                                                                ;RS232 TO CORRESPONDING ONES
                                                                ;FOR THE PRINTER
1918  58                                POP     AX              ;RESTORE AL
1919  0A F6                             OR      DH,DH           ;SEE IF NO ERRORS WERE RETURNED
191B  74 04                             JZ      B12_1
191D  8A E6                             MOV     AH,DH           ;IF THERE WERE ERRORS, RETURN THEM
191F  EB CC                             JMP     SHORT B10_3     ;AND RETURN
1921  B4 10                     B12_1:  MOV     AH,010H         ;PUT 'CORRECT' RETURN STATUS IN AH
1923  EB C8                             JMP     SHORT B10_3     ;AND RETURN
1925                            REPRINT ENDP
                                ;THIS PROC MAPS THE ERRORS RETURNED FROM A BIOS INT14 CALL
                                ;TO THOSE 'LIKE THAT' OF AN INT17 CALL
                                ;BREAK,FRAMING,PARITY,OVERRUN ERRORS ARE LOGGED AS I/O
                                ;ERRORS AND A TIME OUT IS MOVED TO THE APPROPIATE BIT
1925                            FAKE    PROC NEAR
1925  32 F6                             XOR     DH,DH           ;CLEAR FAKED STATUS FLAGS
1927  F6 C4 1E                          TEST    AH,011110B      ;CHECK FOR BREAK,FRAMING,PARITY
                                                                ;OVERRUN
192A  74 03                             JZ      B13_1           ;ERRORS. IF NOT THEN CHECK FOR
                                                                ; TIME OUT.
192C  B6 08                             MOV     DH,01000B       ;SET BIT 3 TO INDICATE 'I/O ERROR'
192E  C3                                RET                     ;AND RETURN
192F  F6 C4 80                  B13_1:  TEST    AH,080H         ;TEST FOR TIME OUT ERROR RETURNED
1932  74 02                             JZ      B13_2           ;IF NOT TIME OUT, RETURN
1934  B6 09                             MOV     DH,09H          ;IF TIME OUT
1936  C3                        B13_2:  RET
1937                            FAKE    ENDP
                                ;--------------------------------------------------------------
                                ;NEW_INT9
                                ;       THIS ROUTINE IS THE INTERRUPT 9 HANDLER WHEN THE MACHINE IS
                                ;       FIRST POWERED ON AND CASSETTE BASIC IS GIVEN CONTROL.  IT
                                ;       HANDLES THE FIRST KEYSTROKES ENTERED FROM THE KEYBOARD AND
                                ;       PERFORMS "SPECIAL" ACTIONS AS FOLLOWS:
                                ;               IF ESC IS THE FIRST KEY ENETERED MINI-WELCOME IS
                                ;                  EXECUTED
                                ;               IF CTRL-ESC IS THE FIRST SEQUENCE "LOAD CAS1:,R" IS
                                ;                  EXECUTED GIVING THE USER THE ABILITY TO BOOT
                                ;                  FROM CASSETTE
                                ;       AFTER THESE KEYSTROKES OR AFTER ANY OTHER KEYSTROKES THE
                                ;       INTERRUPT 9 VECTOR IS CHANGED TO POINT AT THE REAL
                                ;       INTERRUPT 9 ROUTINE.
                                ;--------------------------------------------------------------
1937                            NEW_INT_9  PROC FAR
1937  3C 01                             CMP     AL,1            ; IS THIS AN ESCAPE KEY?
1939  74 10                             JE      ESC_KEY         ; JUMP IF AL=ESCAPE KEY
193B  3C 1D                             CMP     AL,29           ; ELSE, IS THIS A CONTROL KEY?
193D  74 06                             JE      CTRL_KEY        ; JUMP IF AL=CONTROL KEY
193F  E8 E01B R                         CALL    REAL_VECTOR_SETUP ; OTHERWISE, INITIALIZE REAL
                                                                ; INT 9 VECTOR
1942  CD 09                             INT     9H              ; PASS THE SCAN CODE IN AL
1944  CF                                IRET                    ; RETURN TO INTERRUPT 48H
1945                            CTRL_KEY:
1945  80 0E 0017 R 04                   OR      KB_FLAG,04H     ; TURN ON CTRL SHIFT IN KB_FLAG
194A  CF                                IRET                    ; RETURN TO INTERRUPT
194B                            ESC_KEY:
194B  F6 06 0017 R 04                   TEST    KB_FLAG,04H     ; HAS CONTROL SHIFT OCCURED?
1950  74 29                             JE      ESC_ONLY        ; NO.  ESCAPE ONLY
                                ;CONTROL ESCAPE HAS OCCURED, PUT MESSAGE IN BUFFER FOR CASSETTE
                                ; LOAD
1952  C6 06 0017 R 00                   MOV     KB_FLAG,0       ; ZERO OUT CONTROL STATE
1957  1E                                PUSH    DS
1958  07                                POP     ES              ; INITIALIZE ES FOR BIOS DATA
1959  1E                                PUSH    DS              ; SAVE OLD DS
195A  0E                                PUSH    CS              ; POINT DS AT CODE SEGMENT
195B  1F                                POP     DS
195C  BE 1983 R                         MOV     SI,OFFSET CAS_LOAD ; GET MESSAGE
195F  BF 001E R                         MOV     DI,OFFSET KB_BUFFER ; POINT AT KEYBOARD BUFFER
1962  B9 000F 90                        MOV     CX,CAS_LENGTH   ; LENGTH OF CASSETTE MESSAGE
1966  AC                        T_LOOP: LODSB                   ; GET ASCII CHARACTER FROM MESSAGE
1967  AB                                STOSW                   ; PUT IN KEYBOARD BUFFER
1968  E2 FC                             LOOP    T_LOOP
196A  1F                                POP     DS              ; RETRIEVE BIOS DATA SEGMENT
                                ;------ INITIALIZE QUEUE SO MESSAGE WILL BE REMOVED FROM BUFFER
196B  C7 06 001A R 001E R               MOV     BUFFER_HEAD,OFFSET KB_BUFFER
1971  C7 06 001C R 003C R               MOV     BUFFER_TAIL,OFFSET KB_BUFFER+(CAS_LENGTH*2)
                                ;--------------------------------------------------------------
                                ;***NOTE***
                                ;       IT IS ASSUMED THAT THE LENGTH OF THE CASSETTE MESSAGE IS
                                ;       LESS THAN OR EQUAL TO THE LENGTH OF THE BUFFER.  IF THIS IS
                                ;       NOT THE CASE THE BUFFER WILL EVENTUALLY CONSUME MEMORY.
                                ;--------------------------------------------------------------
1977  E8 E01B R                         CALL    REAL_VECTOR_SETUP
197A  CF                                IRET
197B                            ESC_ONLY:
197B  E8 E01B R                         CALL    REAL_VECTOR_SETUP
197E  B9 2000                           MOV     CX,MINI
1981  FF E1                             JMP     CX              ; ENTER THE WORLD OF KEYBOARD CAPER
                                ;------ MESSAGE FOR OUTPUT WHEN CONTROL-ESCAPE IS ENTERED AS FIRST
                                ;       KEY SEQUENCE
1983                            CAS_LOAD        LABEL   BYTE
1983  4C 4F 41 44 20 22                 DB 'LOAD "CAS1:",R'
      43 41 53 31 3A 22
      2C 52
1991  0D                                DB  13
= 000F                          CAS_LENGTH EQU $ - CAS_LOAD
1992                            NEW_INT_9   ENDP
```

```
                                 ;---------------------------------------------
                                 ; WRITE_TTY
                                 ; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
                                 ;   VIDEO CARD.  THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
                                 ;   CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
                                 ;   IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
                                 ;   IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED.  IF THE ROW
                                 ;   ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST
                                 ;   ROW, FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE
                                 ;   LINE. WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING
                                 ;   THE NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE
                                 ;   PREVIOUS LINE BEFORE THE SCROLL, IN CHARACTER MODE.  IN
                                 ;   GRAPHICS MODE, THE 0 COLOR IS USED.
                                 ; ENTRY --
                                 ;   (AH) = CURRENT CRT MODE
                                 ;   (AL) = CHARACTER TO BE WRITTEN
                                 ;          NOTE THAT BACK SPACE, CAR RET, BELL AND LINE FEED ARE
                                 ;          HANDLED AS COMMANDS RATHER THAN AS DISPLAYABLE GRAPHICS
                                 ;   (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A
                                 ;          GRAPHICS MODE
                                 ; EXIT --
                                 ;    ALL REGISTERS SAVED
                                 ;------------------------------------------------
                                         ASSUME  CS:CODE,DS:DATA
1992                             WRITE_TTY       PROC    NEAR
1992  50                                 PUSH    AX                 ; SAVE REGISTERS
1993  50                                 PUSH    AX                 ; SAVE CHAR TO WRITE
1994  8A 3E 0062 R                       MOV     BH,ACTIVE_PAGE     ; GET CURRENT PAGE SETTING
1998  53                                 PUSH    BX                 ; SAVE IT
1999  8A DF                              MOV     BL,BH              ; IN BL
199B  32 FF                              XOR     BH,BH
199D  D1 E3                              SAL     BX,1               ; CONVERT TO WORD OFFSET
199F  8B 97 0050 R                       MOV     DX,[BX+OFFSET CURSOR_POSN] ; GET CURSOR POSITION
19A3  5B                                 POP     BX                 ; RECOVER CURRENT PAGE
19A4  58                                 POP     AX                 ; RECOVER CHAR
                                 ;----- DX NOW HAS THE CURRENT CURSOR POSITION
19A5  3C 08                              CMP     AL,8               ; IS IT A BACKSPACE?
19A7  74 50                              JE      U8                 ; BACK_SPACE
19A9  3C 0D                              CMP     AL,0DH             ; IS IT A CARRIAGE RETURN?
19AB  74 54                              JE      U9                 ; CAR_RET
19AD  3C 0A                              CMP     AL,0AH             ; IS IT A LINE FEED
19AF  74 15                              JE      U10                ; LINE_FEED
19B1  3C 07                              CMP     AL,07H             ; IS IT A BELL
19B3  74 50                              JE      U11                ; BELL
                                 ;----- WRITE THE CHAR TO THE SCREEN
19B5  B4 0A                              MOV     AH,10              ; WRITE CHAR ONLY
19B7  B9 0001                            MOV     CX,1               ; ONLY ONE CHAR
19BA  CD 10                              INT     10H                ; WRITE THE CHAR
                                 ;----- POSITION THE CURSOR FOR NEXT CHAR
19BC  FE C2                              INC     DL
19BE  3A 16 004A R                       CMP     DL,BYTE PTR CRT_COLS ; TEST FOR COLUMN OVERFLOW
19C2  75 31                              JNZ     U7                 ; SET_CURSOR
19C4  32 D2                              XOR     DL,DL              ; COLUMN FOR CURSOR
                                 ;---- LINE FEED
19C6                             U10:
19C6  80 FE 18                           CMP     DH,24
19C9  75 28                              JNZ     U6                 ; SET_CURSOR_INC
                                 ;----- SCROLL REQUIRED
19CB  B4 02                              MOV     AH,2
19CD  CD 10                              INT     10H                ; SET THE CURSOR
                                 ;----- DETERMINE VALUE TO FILL WITH DURING SCROLL
19CF  A0 0049 R                          MOV     AL,CRT_MODE        ; GET THE CURRENT MODE
19D2  3C 04                              CMP     AL,4
19D4  72 04                              JC      U2                 ; READ-CURSOR
19D6  32 FF                              XOR     BH,BH              ; FILL WITH BACKGROUND
19D8  EB 06                              JMP     SHORT U3           ; SCROLL-UP
19DA  B4 08                      U2:     MOV     AH,8
19DC  CD 10                              INT     10H                ; READ CHAR/ATTR AT CURRENT CURSOR
19DE  8A FC                              MOV     BH,AH              ; STORE IN BH
19E0  B8 0601                    U3:     MOV     AX,601H            ; SCROLL ONE LINE
19E3  2B C9                              SUB     CX,CX              ; UPPER LEFT CORNER
19E5  B6 18                              MOV     DH,24              ; LOWER RIGHT ROW
19E7  8A 16 004A R                       MOV     DL,BYTE PTR CRT_COLS ; LOWER RIGHT COLUMN
19EB  FE CA                              DEC     DL
19ED  CD 10                      U4:     INT     10H                ; SCROLL UP THE SCREEN
19EF  58                         U5:     POP     AX                 ; RESTORE THE CHARACTER
19F0  E9 0F70 R                          JMP     VIDEO_RETURN       ; RETURN TO CALLER
19F3  FE C6                      U6:     INC     DH                 ; NEXT ROW
19F5  B4 02                      U7:     MOV     AH,2
19F7  EB F4                              JMP     U4                 ; ESTABLISH THE NEW CURSOR
                                 ;----- BACK SPACE FOUND
19F9  0A D2                      U8:     OR      DL,DL              ; ALREADY AT END OF LINE
19FB  74 F8                              JE      U7                 ; SET_CURSOR
19FD  FE CA                              DEC     DL                 ; NO -- JUST MOVE IT BACK
19FF  EB F4                              JMP     U7                 ; SET_CURSOR
                                 ;----- CARRIAGE RETURN FOUND
1A01  32 D2                      U9:     XOR     DL,DL              ; MOVE TO FIRST COLUMN
1A03  EB F0                              JMP     U7                 ; SET_CURSOR
                                 ;----- BELL FOUND
1A05  B3 02                      U11:    MOV     BL,2               ; SET UP COUNT FOR BEEP
1A07  E8 FF31 R                          CALL    BEEP               ; SOUND THE POD BELL
1A0A  EB E3                              JMP     U5                 ; TTY_RETURN
1A0C                             WRITE_TTY       ENDP
```

```
                                        ;---------------------------------------------------------------
                                        ;       THIS PROCEDURE WILL ISSUE SHORT TONES TO INDICATE FAILURES
                                        ;       THAT  1: OCCUR BEFORE THE CRT IS STARTED, 2: TO CALL THE
                                        ;       OPERATORS ATTENTION TO AN ERROR AT THE END OF POST, OR
                                        ;       3: TO SIGNAL THE SUCCESSFUL COMPLETION OF POST
                                        ; ENTRY PARAMETERS:
                                        ;       DL = NUMBER OF APPROX. 1/2 SEC TONES TO SOUND
                                        ;---------------------------------------------------------------
1A0C                                    ERR_BEEP  PROC  NEAR
1A0C  9C                                        PUSHF                   ; SAVE FLAGS
1A0D  53                                        PUSH    BX
1A0E  FA                                        CLI                     ; DISABLE SYSTEM INTERRUPTS
1A0F                                    G3:                             ; SHORT_BEEP:
1A0F  B3 01                                     MOV     BL,1            ; COUNTER FOR A SHORT BEEP
1A11  E8 FF31 R                                 CALL    BEEP            ; DO THE SOUND
1A14  E2 FE                             G4:     LOOP    G4              ; DELAY BETWEEN BEEPS
1A16  FE CA                                     DEC     DL              ; DONE WITH SHORTS
1A18  75 F5                                     JNZ     G3              ; DO SOME MORE
1A1A  E2 FE                             G5:     LOOP    G5              ; LONG DELAY BEFORE RETURN
1A1C  E2 FE                             G6:     LOOP    G6
1A1E  5B                                        POP     BX              ; RESTORE ORIG CONTENTS OF BX
1A1F  9D                                        POPF                    ; RESTORE FLAGS TO ORIG SETTINGS
1A20  C3                                        RET                     ; RETURN TO CALLER
1A21                                    ERR_BEEP        ENDP
                                         LIST
                                                ASSUME  CS:CODE,DS:DATA
E000                                            ORG     0E000H
E000  31 35 30 34 30 33                         DB      '1504037 COPR. IBM 1981,1983'  ; COPYRIGHT NOTICE
      37 20 43 4F 50 52
      2E 20 49 42 4D 20
      31 39 38 31 2C 31
      39 38 33

                                        ;---------------------------------------------------------------
                                        ;REAL_VECTOR_SETUP
                                        ;
                                        ;       THIS ROUTINE WILL INITIALIZE THE INTERRUPT 9 VECTOR TO
                                        ;       POINT AT THE REAL INTERRUPT ROUTINE.
                                        ;---------------------------------------------------------------
E01B                                    REAL_VECTOR_SETUP       PROC       NEAR
E01B  50                                        PUSH    AX              ; SAVE THE SCAN CODE
E01C  53                                        PUSH    BX
E01D  06                                        PUSH    ES
E01E  33 C0                                     XOR     AX,AX           ; INITIALIZE TO POINT AT VECTOR
                                                                        ; SECTOR(0)
E020  8E C0                                     MOV     ES,AX
E022  BB 0024                                   MOV     BX,9H*4H        ; POINT AT INTERRUPT 9
E025  26: C7 07 1561 R                          MOV     WORD PTR ES:[BX],OFFSET KB_INT ; MOVE IN OFFSET OF
                                                                        ; ROUTINE
E02A  43                                        INC     BX              ; ADD 2 TO BX
E02B  43                                        INC     BX
E02C  0E                                        PUSH    CS              ; GET CODE SEGMENT OF BIOS (SEGMENT
                                                                        ; RELOCATEABLE)
E02D  58                                        POP     AX
E02E  26: 89 07                                 MOV     WORD PTR ES:[BX],AX ; MOVE IN SEGMENT OF ROUTINE
E031  07                                        POP     ES
E032  5B                                        POP     BX
E033  58                                        POP     AX
E034  C3                                        RET
E035                                    REAL_VECTOR_SETUP       ENDP
                                        ;---------------------------------------------------------------
                                        ;KB_NOISE
                                        ;       THIS ROUTINE IS CALLED WHEN GENERAL BEEPS ARE REQUIRED FROM
                                        ;       THE SYSTEM.
                                        ;INPUT
                                        ;       BX=LENGH OF THE TONE
                                        ;       CX=CONTAINS THE FREQUENCY
                                        ;OUTPUT
                                        ;       ALL REGISTERS ARE MAINTAINED.
                                        ;HINTS
                                        ;       AS CX GETS LARGER THE TONE PRODUCED GETS LOWER IN PITCH.
                                        ;
                                        ;---------------------------------------------------------------
E035                                    KB_NOISE        PROC    NEAR
E035  FB                                        STI
E036  50                                        PUSH    AX
E037  53                                        PUSH    BX
E038  51                                        PUSH    CX
E039  E4 61                                     IN      AL,061H         ; GET CONTROL INFO
E03B  50                                        PUSH    AX              ; SAVE
E03C                                    LOOP01:
E03C  24 FC                                     AND     AL,0FCH         ; TURN OFF TIMER GATE AND SPEAKER
                                                                        ; DATA
E03E  E6 61                                     OUT     061H,AL         ; OUTPUT TO CONTROL
E040  51                                        PUSH    CX              ; HALF CYCLE TIME FOR TONE
E041  E2 FE                             LOOP02: LOOP    LOOP02          ; SPEAKER OFF
E043  0C 02                                     OR      AL,2            ; TURN ON SPEAKER BIT
E045  E6 61                                     OUT     061H,AL         ; OUTPUT TO CONTROL
E047  59                                        POP     CX
E048  51                                        PUSH    CX              ; RETRIEVE FREQUENCY
E049  E2 FE                             LOOP03: LOOP    LOOP03          ; ANOTHER HALF CYCLE
E04B  4B                                        DEC     BX              ; TOTAL TIME COUNT
E04C  59                                        POP     CX              ; RETRIEVE  FREQ.
E04D  75 ED                                     JNZ     LOOP01          ; DO ANOTHER CYCLE
E04F  58                                        POP     AX              ; RECOVER CONTROL
E050  E6 61                                     OUT     061H,AL         ; OUTPUT THE CONTROL
E052  59                                        POP     CX
E053  5B                                        POP     BX
E054  58                                        POP     AX
E055  C3                                        RET
E056                                    KB_NOISE           ENDP
E05B                                            ORG     0E05BH
E05B  E9 0043 R                                 JMP     NEAR PTR RESET
```

```
                                        ;-----------------------------------------------------------
                                        ;       CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200
                                        ;       GRAPHICS FOR CHARACTERS 80H THROUGH FFH
                                        ;-----------------------------------------------------------

E05E                                    CRT_CHARH       LABEL   BYTE
E05E   78 CC C0 CC 78 18                        DB      078H,0CCH,0C0H,0CCH,078H,018H,00CH,078H  ; D_80
       0C 78
E066   00 CC 00 CC CC CC                        DB      000H,0CCH,000H,0CCH,0CCH,0CCH,07EH,000H  ; D_81
       7E 00
E06E   1C 00 78 CC FC C0                        DB      01CH,000H,078H,0CCH,0FCH,0C0H,078H,000H  ; D_82
       78 00
E076   7E C3 3C 06 3E 66                        DB      07EH,0C3H,03CH,006H,03EH,066H,03FH,000H  ; D_83
       3F 00
E07E   CC 00 78 0C 7C CC                        DB      0CCH,000H,078H,00CH,07CH,0CCH,07EH,000H  ; D_84
       7E 00
E086   E0 00 78 0C 7C CC                        DB      0E0H,000H,078H,00CH,07CH,0CCH,07EH,000H  ; D_85
       7E 00
E08E   30 30 78 0C 7C CC                        DB      030H,030H,078H,00CH,07CH,0CCH,07EH,000H  ; D_86
       7E 00
E096   00 00 78 C0 C0 78                        DB      000H,000H,078H,0C0H,0C0H,078H,00CH,038H  ; D_87
       0C 38
E09E   7E C3 3C 66 7E 60                        DB      07EH,0C3H,03CH,066H,07EH,060H,03CH,000H  ; D_88
       3C 00
E0A6   CC 00 78 CC FC C0                        DB      0CCH,000H,078H,0CCH,0FCH,0C0H,078H,000H  ; D_89
       78 00
E0AE   E0 00 78 CC FC C0                        DB      0E0H,000H,078H,0CCH,0FCH,0C0H,078H,000H  ; D_8A
       78 00
E0B6   CC 00 70 30 30 30                        DB      0CCH,000H,070H,030H,030H,030H,078H,000H  ; D_8B
       78 00
E0BE   7C C6 38 18 18 18                        DB      07CH,0C6H,038H,018H,018H,018H,03CH,000H  ; D_8C
       3C 00
E0C6   E0 00 70 30 30 30                        DB      0E0H,000H,070H,030H,030H,030H,078H,000H  ; D_8D
       78 00
E0CE   C6 38 6C C6 FE C6                        DB      0C6H,038H,06CH,0C6H,0FEH,0C6H,0C6H,000H  ; D_8E
       C6 00
E0D6   30 30 00 78 CC FC                        DB      030H,030H,000H,078H,0CCH,0FCH,0CCH,000H  ; D_8F
       CC 00

E0DE   1C 00 FC 60 78 60                        DB      01CH,000H,0FCH,060H,078H,060H,0FCH,000H  ; D_90
       FC 00
E0E6   00 00 7F 0C 7F CC                        DB      000H,000H,07FH,00CH,07FH,0CCH,07FH,000H  ; D_91
       7F 00
E0EE   3E 6C CC FE CC CC                        DB      03EH,06CH,0CCH,0FEH,0CCH,0CCH,0CEH,000H  ; D_92
       CE 00
E0F6   78 CC 00 78 CC CC                        DB      078H,0CCH,000H,078H,0CCH,0CCH,078H,000H  ; D_93
       78 00
E0FE   00 CC 00 78 CC CC                        DB      000H,0CCH,000H,078H,0CCH,0CCH,078H,000H  ; D_94
       78 00
E106   00 E0 00 78 CC CC                        DB      000H,0E0H,000H,078H,0CCH,0CCH,078H,000H  ; D_95
       78 00
E10E   78 CC 00 CC CC CC                        DB      078H,0CCH,000H,0CCH,0CCH,0CCH,07EH,000H  ; D_96
       7E 00
E116   00 E0 00 CC CC CC                        DB      000H,0E0H,000H,0CCH,0CCH,0CCH,07EH,000H  ; D_97
       7E 00
E11E   00 CC 00 CC CC 7C                        DB      000H,0CCH,000H,0CCH,0CCH,07CH,00CH,0F8H  ; D_98
       0C F8
E126   C3 18 3C 66 66 3C                        DB      0C3H,018H,03CH,066H,066H,03CH,018H,000H  ; D_99
       18 00
E12E   CC 00 CC CC CC CC                        DB      0CCH,000H,0CCH,0CCH,0CCH,0CCH,078H,000H  ; D_9A
       78 00
E136   18 18 7E C0 C0 7E                        DB      018H,018H,07EH,0C0H,0C0H,07EH,018H,018H  ; D_9B
       18 18
E13E   38 6C 64 F0 60 E6                        DB      038H,06CH,064H,0F0H,060H,0E6H,0FCH,000H  ; D_9C
       FC 00
E146   CC CC 78 FC 30 FC                        DB      0CCH,0CCH,078H,0FCH,030H,0FCH,030H,030H  ; D_9D
       30 30
E14E   F8 CC CC FA C6 CF                        DB      0F8H,0CCH,0CCH,0FAH,0C6H,0CFH,0C6H,0C7H  ; D_9E
       C6 C7
E156   0E 1B 18 3C 18 18                        DB      00EH,01BH,018H,03CH,018H,018H,0D8H,070H  ; D_9F
       D8 70

E15E   1C 00 78 0C 7C CC                        DB      01CH,000H,078H,00CH,07CH,0CCH,07EH,000H  ; D_A0
       7E 00
E166   38 00 70 30 30 30                        DB      038H,000H,070H,030H,030H,030H,078H,000H  ; D_A1
       78 00
E16E   00 1C 00 78 CC CC                        DB      000H,01CH,000H,078H,0CCH,0CCH,078H,000H  ; D_A2
       78 00
E176   00 1C 00 CC CC CC                        DB      000H,01CH,000H,0CCH,0CCH,0CCH,07EH,000H  ; D_A3
       7E 00
E17E   00 F8 00 F8 CC CC                        DB      000H,0F8H,000H,0F8H,0CCH,0CCH,0CCH,000H  ; D_A4
       CC 00
E186   FC 00 CC EC FC DC                        DB      0FCH,000H,0CCH,0ECH,0FCH,0DCH,0CCH,000H  ; D_A5
       CC 00
E18E   3C 6C 6C 3E 00 7E                        DB      03CH,06CH,06CH,03EH,000H,07EH,000H,000H  ; D_A6
       00 00
E196   38 6C 6C 38 00 7C                        DB      038H,06CH,06CH,038H,000H,07CH,000H,000H  ; D_A7
       00 00
E19E   30 00 30 60 C0 CC                        DB      030H,000H,030H,060H,0C0H,0CCH,078H,000H  ; D_A8
       78 00
E1A6   00 00 00 FC C0 C0                        DB      000H,000H,000H,0FCH,0C0H,0C0H,000H,000H  ; D_A9
       00 00
E1AE   00 00 00 FC 0C 0C                        DB      000H,000H,000H,0FCH,00CH,00CH,000H,000H  ; D_AA
       00 00
E1B6   C3 C6 CC DE 33 66                        DB      0C3H,0C6H,0CCH,0DEH,033H,066H,0CCH,00FH  ; D_AB
       CC 0F
E1BE   C3 C6 CC DB 37 6F                        DB      0C3H,0C6H,0CCH,0DBH,037H,06FH,0CFH,003H  ; D_AC
       CF 03
E1C6   18 18 00 18 18 18                        DB      018H,018H,000H,018H,018H,018H,018H,000H  ; D_AD
       18 00
E1CE   00 33 66 CC 66 33                        DB      000H,033H,066H,0CCH,066H,033H,000H,000H  ; D_AE
       00 00
E1D6   00 CC 66 33 66 CC                        DB      000H,0CCH,066H,033H,066H,0CCH,000H,000H  ; D_AF
       00 00
```

```
E1DE  22 88 22 88 22 88         DB      022H,088H,022H,088H,022H,088H,022H,088H  ; D_B0
      22 88
E1E6  55 AA 55 AA 55 AA         DB      055H,0AAH,055H,0AAH,055H,0AAH,055H,0AAH  ; D_B1
      55 AA
E1EE  DB 77 DB EE DB 77         DB      0DBH,077H,0DBH,0EEH,0DBH,077H,0DBH,0EEH  ; D_B2
      DB EE
E1F6  18 18 18 18 18 18         DB      018H,018H,018H,018H,018H,018H,018H,018H  ; D_B3
      18 18
E1FE  18 18 18 18 F8 18         DB      018H,018H,018H,018H,0F8H,018H,018H,018H  ; D_B4
      18 18
E206  18 18 F8 18 F8 18         DB      018H,018H,0F8H,018H,0F8H,018H,018H,018H  ; D_B5
      18 18
E20E  36 36 36 36 F6 36         DB      036H,036H,036H,036H,0F6H,036H,036H,036H  ; D_B6
      36 36
E216  00 00 00 00 FE 36         DB      000H,000H,000H,000H,0FEH,036H,036H,036H  ; D_B7
      36 36
E21E  00 00 F8 18 F8 18         DB      000H,000H,0F8H,018H,0F8H,018H,018H,018H  ; D_B8
      18 18
E226  36 36 F6 06 F6 36         DB      036H,036H,0F6H,006H,0F6H,036H,036H,036H  ; D_B9
      36 36
E22E  36 36 36 36 36 36         DB      036H,036H,036H,036H,036H,036H,036H,036H  ; D_BA
      36 36
E236  00 00 FE 06 F6 36         DB      000H,000H,0FEH,006H,0F6H,036H,036H,036H  ; D_BB
      36 36
E23E  36 36 F6 06 FE 00         DB      036H,036H,0F6H,006H,0FEH,000H,000H,000H  ; D_BC
      00 00
E246  36 36 36 36 FE 00         DB      036H,036H,036H,036H,0FEH,000H,000H,000H  ; D_BD
      00 00
E24E  18 18 F8 18 F8 00         DB      018H,018H,0F8H,018H,0F8H,000H,000H,000H  ; D_BE
      00 00
E256  00 00 00 00 F8 18         DB      000H,000H,000H,000H,0F8H,018H,018H,018H  ; D_BF
      18 18

E25E  18 18 18 18 1F 00         DB      018H,018H,018H,018H,01FH,000H,000H,000H  ; D_C0
      00 00
E266  18 18 18 18 FF 00         DB      018H,018H,018H,018H,0FFH,000H,000H,000H  ; D_C1
      00 00
E26E  00 00 00 00 FF 18         DB      000H,000H,000H,000H,0FFH,018H,018H,018H  ; D_C2
      18 18
E276  18 18 18 18 1F 18         DB      018H,018H,018H,018H,01FH,018H,018H,018H  ; D_C3
      18 18
E27E  00 00 00 00 FF 00         DB      000H,000H,000H,000H,0FFH,000H,000H,000H  ; D_C4
      00 00
E286  18 18 18 18 FF 18         DB      018H,018H,018H,018H,0FFH,018H,018H,018H  ; D_C5
      18 18
E28E  18 18 1F 18 1F 18         DB      018H,018H,01FH,018H,01FH,018H,018H,018H  ; D_C6
      18 18
E296  36 36 36 36 37 36         DB      036H,036H,036H,036H,037H,036H,036H,036H  ; D_C7
      36 36
E29E  36 36 37 30 3F 00         DB      036H,036H,037H,030H,03FH,000H,000H,000H  ; D_C8
      00 00
E2A6  00 00 3F 30 37 36         DB      000H,000H,03FH,030H,037H,036H,036H,036H  ; D_C9
      36 36
E2AE  36 36 F7 00 FF 00         DB      036H,036H,0F7H,000H,0FFH,000H,000H,000H  ; D_CA
      00 00
E2B6  00 00 FF 00 F7 36         DB      000H,000H,0FFH,000H,0F7H,036H,036H,036H  ; D_CB
      36 36
E2BE  36 36 37 30 37 36         DB      036H,036H,037H,030H,037H,036H,036H,036H  ; D_CC
      36 36
E2C6  00 00 FF 00 FF 00         DB      000H,000H,0FFH,000H,0FFH,000H,000H,000H  ; D_CD
      00 00
E2CE  36 36 F7 00 F7 36         DB      036H,036H,0F7H,000H,0F7H,036H,036H,036H  ; D_CE
      36 36
E2D6  18 18 FF 00 FF 00         DB      018H,018H,0FFH,000H,0FFH,000H,000H,000H  ; D_CF
      00 00

E2DE  36 36 36 36 FF 00         DB      036H,036H,036H,036H,0FFH,000H,000H,000H  ; D_D0
      00 00
E2E6  00 00 FF 00 FF 18         DB      000H,000H,0FFH,000H,0FFH,018H,018H,018H  ; D_D1
      18 18
E2EE  00 00 00 00 FF 36         DB      000H,000H,000H,000H,0FFH,036H,036H,036H  ; D_D2
      36 36
E2F6  36 36 36 36 3F 00         DB      036H,036H,036H,036H,03FH,000H,000H,000H  ; D_D3
      00 00
E2FE  18 18 1F 18 1F 00         DB      018H,018H,01FH,018H,01FH,000H,000H,000H  ; D_D4
      00 00
E306  00 00 1F 18 1F 18         DB      000H,000H,01FH,018H,01FH,018H,018H,018H  ; D_D5
      18 18
E30E  00 00 00 00 3F 36         DB      000H,000H,000H,000H,03FH,036H,036H,036H  ; D_D6
      36 36
E316  36 36 36 36 FF 36         DB      036H,036H,036H,036H,0FFH,036H,036H,036H  ; D_D7
      36 36
E31E  18 18 FF 18 FF 18         DB      018H,018H,0FFH,018H,0FFH,018H,018H,018H  ; D_D8
      18 18
E326  18 18 18 18 F8 00         DB      018H,018H,018H,018H,0F8H,000H,000H,000H  ; D_D9
      00 00
E32E  00 00 00 00 1F 18         DB      000H,000H,000H,000H,01FH,018H,018H,018H  ; D_DA
      18 18
E336  FF FF FF FF FF FF         DB      0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH  ; D_DB
      FF FF
E33E  00 00 00 00 FF FF         DB      000H,000H,000H,000H,0FFH,0FFH,0FFH,0FFH  ; D_DC
      FF FF
E346  F0 F0 F0 F0 F0 F0         DB      0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H  ; D_DD
      F0 F0
E34E  0F 0F 0F 0F 0F 0F         DB      00FH,00FH,00FH,00FH,00FH,00FH,00FH,00FH  ; D_DE
      0F 0F
E356  FF FF FF FF 00 00         DB      0FFH,0FFH,0FFH,0FFH,000H,000H,000H,000H  ; D_DF
      00 00
```

```
E35E  00 00 76 DC C8 DC                DB     000H,000H,076H,0DCH,0C8H,0DCH,076H,000H  ; D_E0
      76 00
E366  00 78 CC F8 CC F8                DB     000H,078H,0CCH,0F8H,0CCH,0F8H,0C0H,0C0H  ; D_E1
      C0 C0
E36E  00 FC CC C0 C0 C0                DB     000H,0FCH,0CCH,0C0H,0C0H,0C0H,0C0H,000H  ; D_E2
      C0 00
E376  00 FE 6C 6C 6C 6C                DB     000H,0FEH,06CH,06CH,06CH,06CH,06CH,000H  ; D_E3
      6C 00
E37E  FC CC 60 30 60 CC                DB     0FCH,0CCH,060H,030H,060H,0CCH,0FCH,000H  ; D_E4
      FC 00
E386  00 00 7E D8 D8 D8                DB     000H,000H,07EH,0D8H,0D8H,0D8H,070H,000H  ; D_E5
      70 00
E38E  00 66 66 66 66 7C                DB     000H,066H,066H,066H,066H,07CH,060H,0C0H  ; D_E6
      60 C0
E396  00 76 DC 18 18 18                DB     000H,076H,0DCH,018H,018H,018H,018H,000H  ; D_E7
      18 00
E39E  FC 30 78 CC CC 78                DB     0FCH,030H,078H,0CCH,0CCH,078H,030H,0FCH  ; D_E8
      30 FC
E3A6  38 6C C6 FE C6 6C                DB     038H,06CH,0C6H,0FEH,0C6H,06CH,038H,000H  ; D_E9
      38 00
E3AE  38 6C C6 C6 6C 6C                DB     038H,06CH,0C6H,0C6H,06CH,06CH,0EEH,000H  ; D_EA
      EE 00
E3B6  1C 30 18 7C CC CC                DB     01CH,030H,018H,07CH,0CCH,0CCH,078H,000H  ; D_EB
      78 00
E3BE  00 00 7E DB DB 7E                DB     000H,000H,07EH,0DBH,0DBH,07EH,000H,000H  ; D_EC
      00 00
E3C6  06 0C 7E DB DB 7E                DB     006H,00CH,07EH,0DBH,0DBH,07EH,060H,0C0H  ; D_ED
      60 C0
E3CE  38 60 C0 F8 C0 60                DB     038H,060H,0C0H,0F8H,0C0H,060H,038H,000H  ; D_EE
      38 00
E3D6  78 CC CC CC CC CC                DB     078H,0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,000H  ; D_EF
      CC 00

E3DE  00 FC 00 FC 00 FC                DB     000H,0FCH,000H,0FCH,000H,0FCH,000H,000H  ; D_F0
      00 00
E3E6  30 30 FC 30 30 00                DB     030H,030H,0FCH,030H,030H,000H,0FCH,000H  ; D_F1
      FC 00
E3EE  60 30 18 30 60 00                DB     060H,030H,018H,030H,060H,000H,0FCH,000H  ; D_F2
      FC 00
E3F6  18 30 60 30 18 00                DB     018H,030H,060H,030H,018H,000H,0FCH,000H  ; D_F3
      FC 00
E3FE  0E 1B 1B 18 18 18                DB     00EH,01BH,01BH,018H,018H,018H,018H,018H  ; D_F4
      18 18
E406  18 18 18 18 18 D8                DB     018H,018H,018H,018H,018H,0D8H,0D8H,070H  ; D_F5
      D8 70
E40E  30 30 00 FC 00 30                DB     030H,030H,000H,0FCH,000H,030H,030H,000H  ; D_F6
      30 00
E416  00 76 DC 00 76 DC                DB     000H,076H,0DCH,000H,076H,0DCH,000H,000H  ; D_F7
      00 00
E41E  38 6C 6C 38 00 00                DB     038H,06CH,06CH,038H,000H,000H,000H,000H  ; D_F8
      00 00
E426  00 00 00 18 18 00                DB     000H,000H,000H,018H,018H,000H,000H,000H  ; D_F9
      00 00
E42E  00 00 00 00 18 00                DB     000H,000H,000H,000H,018H,000H,000H,000H  ; D_FA
      00 00
E436  0F 0C 0C 0C EC 6C                DB     00FH,00CH,00CH,00CH,0ECH,06CH,03CH,01CH  ; D_FB
      3C 1C
E43E  78 6C 6C 6C 6C 00                DB     078H,06CH,06CH,06CH,06CH,000H,000H,000H  ; D_FC
      00 00
E446  70 18 30 60 78 00                DB     070H,018H,030H,060H,078H,000H,000H,000H  ; D_FD
      00 00
E44E  00 00 3C 3C 3C 3C                DB     000H,000H,03CH,03CH,03CH,03CH,000H,000H  ; D_FE
      00 00
E456  00 00 00 00 00 00                DB     000H,000H,000H,000H,000H,000H,000H,000H  ; D_FF
      00 00
                                       ASSUME CS:CODE,DS:DATA
                               ;--------------------------------------------------
                               ; SET_CTYPE
                               ;       THIS ROUTINE SETS THE CURSOR VALUE
                               ; INPUT
                               ;       (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
                               ; OUTPUT
                               ;       NONE
                               ;---------------------------------------------------
E45E                           SET_CTYPE       PROC    NEAR
E45E  80 FC 04                         CMP     AH,4            ; IN GRAPHICS MODE?
E461  72 03                            JC      C23X            ; NO, JUMP
E463  80 CD 20                         OR      CH,20H          ; YES, DISABLE CURSOR
E466  B4 0A                    C23X:   MOV     AH,10           ; 6845 REGISTER FOR CURSOR SET
E468  89 0E 0060 R                     MOV     CURSOR_MODE,CX  ; SAVE IN DATA AREA
E46C  E8 E472 R                        CALL    C23             ; OUTPUT CX REG
E46F  E9 0F70 R                        JMP     VIDEO_RETURN
                               ;THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGS NAMED IN AH
E472  8B 16 0063 R             C23:    MOV     DX,ADDR_6845    ; ADDRESS REGISTER
E476  8A C4                            MOV     AL,AH           ; GET VALUE
E478  EE                               OUT     DX,AL           ; REGISTER SET
E479  42                               INC     DX              ; DATA REGISTER
E47A  8A C5                            MOV     AL,CH           ; DATA
E47C  EE                               OUT     DX,AL
E47D  4A                               DEC     DX
E47E  8A C4                            MOV     AL,AH
E480  FE C0                            INC     AL              ; POINT TO OTHER DATA REGISTER
E482  EE                               OUT     DX,AL           ; SET FOR SECOND REGISTER
E483  42                               INC     DX
E484  8A C1                            MOV     AL,CL           ; SECOND DATA VALUE
E486  EE                               OUT     DX,AL
E487  C3                               RET                     ; ALL DONE
E488                           SET_CTYPE       ENDP
```

```
                              ;----------------------------------------------
                              ; SET_CPOS
                              ;       THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
                              ;       NEW X-Y VALUES PASSED
                              ; INPUT
                              ;       DX - ROW,COLUMN OF NEW CURSOR
                              ;       BH - DISPLAY PAGE OF CURSOR
                              ; OUTPUT
                              ;       CURSOR IS SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
                              ;----------------------------------------------
E488                          SET_CPOS        PROC    NEAR
E488  8A CF                           MOV     CL,BH
E48A  32 ED                           XOR     CH,CH           ; ESTABLISH LOOP COUNT
E48C  D1 E1                           SAL     CX,1            ; WORD OFFSET
E48E  8B F1                           MOV     SI,CX           ; USE INDEX REGISTER
E490  89 94 0050 R                    MOV     [SI+OFFSET CURSOR_POSN],DX ; SAVE THE POINTER
E494  38 3E 0062 R                    CMP     ACTIVE_PAGE,BH
E498  75 05                           JNZ     C24             ; SET_CPOS_RETURN
E49A  8B C2                           MOV     AX,DX           ; GET ROW/COLUMN TO AX
E49C  E8 E4A2 R                       CALL    C25             ; CURSOR_SET
E49F  E9 0F70 R               C24:    JMP     VIDEO_RETURN
E4A2                          SET_CPOS        ENDP
                              ;------ SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
E4A2                          C25     PROC    NEAR
E4A2  E8 E5C2 R                       CALL    POSITION        ; DETERMINE LOCATION IN REGEN
                                                              ; BUFFER
E4A5  8B C8                           MOV     CX,AX
E4A7  03 0E 004E R                    ADD     CX,CRT_START    ; ADD IN THE START ADDRESS FOR THIS
                                                              ; PAGE
E4AB  D1 F9                           SAR     CX,1            ; DIVIDE BY 2 FOR CHAR ONLY COUNT
E4AD  B4 0E                           MOV     AH,14           ; REGISTER NUMBER FOR CURSOR
E4AF  E8 E472 R                       CALL    C23             ; OUTPUT THE VALUE TO THE 6845
E4B2  C3                              RET
E4B3                          C25     ENDP
                              ;----------------------------------------------
                              ; ACT_DISP_PAGE
                              ;       THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
                              ;       THE FULL USE OF THE RAM SET ASIDE FOR THE VIDEO ATTACHMENT
                              ; INPUT
                              ;       AL HAS THE NEW ACTIVE DISPLAY PAGE
                              ; OUTPUT
                              ;       THE 6845 IS RESET TO DISPLAY THAT PAGE
                              ;----------------------------------------------
E4B3                          ACT_DISP_PAGE   PROC    NEAR
E4B3  A8 80                           TEST    AL,080H         ; CRT/CPU PAGE REG FUNCTION
E4B5  75 24                           JNZ     SET_CRTCPU      ; YES, GO HANDLE IT
E4B7  A2 0062 R                       MOV     ACTIVE_PAGE,AL  ; SAVE ACTIVE PAGE VALUE
E4BA  8B 0E 004C R                    MOV     CX,CRT_LEN      ; GET SAVED LENGTH OF REGEN BUFFER
E4BE  98                              CBW                     ; CONVERT AL TO WORD
E4BF  50                              PUSH    AX              ; SAVE PAGE VALUE
E4C0  F7 E1                           MUL     CX              ; DISPLAY PAGE TIMES REGEN LENGTH
E4C2  A3 004E R                       MOV     CRT_START,AX    ; SAVE START ADDRESS FOR LATER USE
E4C5  8B C8                           MOV     CX,AX           ; START ADDRESS TO CX
E4C7  D1 F9                           SAR     CX,1            ; DIVIDE BY 2 FOR 6845 HANDLING
E4C9  B4 0C                           MOV     AH,12           ; 6845 REGISTER FOR START ADDRESS
E4CB  E8 E472 R                       CALL    C23
E4CE  5B                              POP     BX              ; RECOVER PAGE VALUE
E4CF  D1 E3                           SAL     BX,1            ; *2 FOR WORD OFFSET
E4D1  8B 87 0050 R                    MOV     AX,[BX + OFFSET CURSOR_POSN] ; GET CURSOR FOR THIS
                                                              ; PAGE
E4D5  E8 E4A2 R                       CALL    C25             ; SET THE CURSOR POSITION
E4D8  E9 0F70 R                       JMP     VIDEO_RETURN
                              ;----------------------------------------------
                              ; SET_CRTCPU
                              ;       THIS ROUTINE READS OR WRITES THE CRT/CPU PAGE REGISTERS
                              ;
                              ; INPUT
                              ;       (AL) = 83H      SET BOTH CRT AND CPU PAGE REGS
                              ;               (BH) =  VALUE TO SET IN CRT PAGE REG
                              ;               (BL) =  VALUE TO SET IN CPU PAGE REG
                              ;       (AL) = 82H      SET CRT PAGE REG
                              ;               (BH) =  VALUE TO SET IN CRT PAGE REG
                              ;       (AL) = 81H      SET CPU PAGE REG
                              ;               (BL) =  VALUE TO SET IN CPU PAGE REG
                              ;       (AL) = 80H      READ CURRENT VALUE OF CRT/CPU PAGE REGS
                              ;
                              ; OUTPUT
                              ;       ALL FUNCTIONS RETURN
                              ;               (BH) = CURRENT CONTENTS OF CRT PAGE REG
                              ;               (BL) = CURRENT CONTENTS OF CPU PAGE REG
                              ;----------------------------------------------
E4DB                          SET_CRTCPU:
E4DB  8A E0                           MOV     AH,AL           ; SAVE REQUEST IN AH
E4DD  BA 03DA                         MOV     DX,VGA_CTL      ; SET ADDRESS OF GATE ARRAY
E4E0  EC                      C26:    IN      AL,DX           ; GET STATUS
E4E1  24 08                           AND     AL,08H          ; VERTICAL RETRACE?
E4E3  74 FB                           JZ      C26             ; NO, WAIT FOR IT
E4E5  BA 03DF                         MOV     DX,PAGREG       ; SET IO ADDRESS OF PAGE REG
E4E8  A0 008A R                       MOV     AL,PAGDAT       ; GET DATA LAST OUTPUT TO REG
E4EB  80 FC 80                        CMP     AH,80H          ; READ FUNCTION REQUESTED?
E4EE  74 27                           JZ      C29             ; YES, DON'T SET ANYTHING
E4F0  80 FC 84                        CMP     AH,84H          ; VALID REQUEST?
E4F3  73 22                           JNC     C29             ; NO, PRETEND IT WAS A READ REQUEST
E4F5  F6 C4 01                        TEST    AH,1            ; SET CPU REG?
E4F8  74 0D                           JZ      C27             ; NO, GO SEE ABOUT CRT REG
E4FA  D0 E3                           SHL     BL,1            ; SHIFT VALUE TO RIGHT BIT POSITION
E4FC  D0 E3                           SHL     BL,1
E4FE  D0 E3                           SHL     BL,1
E500  24 C7                           AND     AL,NOT CPUREG   ; CLEAR OLD CPU VALUE
E502  80 E3 38                        AND     BL,CPUREG       ; BE SURE UNRELATED BITS ARE ZERO
E505  0A C3                           OR      AL,BL           ; OR IN NEW VALUE
```

```
E507  F6 C4 02                  C27:    TEST    AH,2              ; SET CRT REG?
E50A  74 07                             JZ      C28               ; NO, GO RETURN CURRENT SETTINGS
E50C  24 F8                             AND     AL,NOT CRTREG     ; CLEAR OLD CRT VALUE
E50E  80 E7 07                          AND     BH,CRTREG         ; BE SURE UNRELATED BITS ARE ZERO
E511  0A C7                             OR      AL,BH             ; OR IN NEW VALUE
E513  EE                        C28:    OUT     DX,AL             ; SET NEW VALUES
E514  A2 008A R                         MOV     PAGDAT,AL         ; SAVE COPY IN RAM
E517  8A D8                     C29:    MOV     BL,AL             ; GET CPU REG VALUE
E519  80 E3 38                          AND     BL,CPUREG         ; CLEAR EXTRA BITS
E51C  D0 FB                             SAR     BL,1              ; RIGHT JUSTIFY IN BL
E51E  D0 FB                             SAR     BL,1
E520  D0 FB                             SAR     BL,1
E522  8A F8                             MOV     BH,AL             ; GET CRT REG VALUE
E524  80 E7 07                          AND     BH,CRTREG         ; CLEAR EXTRA BITS
E527  5F                                POP     DI                ; RESTORE SOME REGS
E528  5E                                POP     SI
E529  58                                POP     AX                ; DISCARD SAVED BX
E52A  E9 0F73 R                         JMP     C22               ; RETURN
E52D                            ACT_DISP_PAGE   ENDP
                                ;-----------------------------------------------
                                ; READ_CURSOR
                                ;       THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
                                ;       6845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
                                ; INPUT
                                ;       BH - PAGE OF CURSOR
                                ; OUTPUT
                                ;       DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
                                ;       CX - CURRENT CURSOR MODE
                                ;-----------------------------------------
E52D                            READ_CURSOR     PROC    NEAR
E52D  8A DF                             MOV     BL,BH
E52F  32 FF                             XOR     BH,BH
E531  D1 E3                             SAL     BX,1              ; WORD OFFSET
E533  8B 97 0050 R                      MOV     DX,[BX+OFFSET CURSOR_POSN]
E537  8B 0E 0060 R                      MOV     CX,CURSOR_MODE
E53B  5F                                POP     DI
E53C  5E                                POP     SI
E53D  5B                                POP     BX
E53E  58                                POP     AX                ; DISCARD SAVED CX AND DX
E53F  58                                POP     AX
E540  1F                                POP     DS
E541  07                                POP     ES
E542  CF                                IRET
E543                            READ_CURSOR     ENDP
                                ;-------------------------------------------
                                ; SET COLOR
                                ;       THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE
                                ;       OVERSCAN COLOR, AND THE FOREGROUND COLOR SET FOR GRAPHICS
                                ; INPUT
                                ;       (BH) HAS COLOR ID
                                ;               IF BH=0, THE BACKGROUND COLOR VALUE IS SET
                                ;                       FROM THE LOW BITS OF BL (0-31)
                                ;                       IN GRAPHIC MODES, BOTH THE BACKGROUND AND
                                ;                       BORDER ARE SET. IN ALPHA MODES, ONLY THE
                                ;                       BORDER IS SET.
                                ;               IF BH=1, THE PALETTE SELECTION IS MADE
                                ;                       BASED ON THE LOW BIT OF BL:
                                ;                          2 COLOR MODE:
                                ;                               0 = WHITE FOR COLOR 1
                                ;                               1 = BLACK FOR COLOR 1
                                ;                          4 COLOR MODES:
                                ;                               0 = GREEN, RED, YELLOW FOR
                                ;                                   COLORS 1,2,3
                                ;                               1 = BLUE, CYAN, MAGENTA FOR
                                ;                                   COLORS 1,2,3
                                ;                          16 COLOR MODES:
                                ;                               ALWAYS SETS UP PALETTE AS:
                                ;                               BLUE FOR COLOR 1
                                ;                               GREEN FOR COLOR 2
                                ;                               CYAN FOR COLOR 3
                                ;                               RED FOR COLOR 4
                                ;                               MAGENTA FOR COLOR 5
                                ;                               BROWN FOR COLOR 6
                                ;                               LIGHT GRAY FOR COLOR 7
                                ;                               DARK GRAY FOR COLOR 8
                                ;                               LIGHT BLUE FOR COLOR 9
                                ;                               LIGHT GREEN FOR COLOR 10
                                ;                               LIGHT CYAN FOR COLOR 11
                                ;                               LIGHT RED FOR COLOR 12
                                ;                               LIGHT MAGENTA FOR COLOR 13
                                ;                               YELLOW FOR COLOR 14
                                ;                               WHITE FOR COLOR 15
                                ;       (BL) HAS THE COLOR VALUE TO BE USED
                                ; OUTPUT
                                ;       THE COLOR SELECTION IS UPDATED
                                ;-------------------------------------------
E543                            SET_COLOR       PROC    NEAR
E543  BA 03DA                           MOV     DX,VGA_CTL        ; I/O PORT FOR PALETTE
E546  EC                        C30:    IN      AL,DX             ; SYNC UP VGA FOR REG ADDRESS
E547  A8 08                             TEST    AL,8              ; IS VERTICAL RETRACE ON?
E549  74 FB                             JZ      C30               ; NO, WAIT UNTIL IT IS
E54B  0A FF                             OR      BH,BH             ; IS THIS COLOR 0?
E54D  75 19                             JNZ     C31               ; OUTPUT COLOR 1
```

```
                                         ;------ HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
                                         ;              AND BORDER COLOR
E54F  80 3E 0049 R 04                           CMP     CRT_MODE,4         ; IN ALPHA MODE?
E554  72 06                                     JC      C305               ; YES, JUST SET BORDER REG
E556  B0 10                                     MOV     AL,10H             ; SET PALETTE REG 0
E558  EE                                        OUT     DX,AL              ; SELECT VGA REG
E559  8A C3                                     MOV     AL,BL              ; GET COLOR
E55B  EE                                        OUT     DX,AL              ; SET IT
E55C  B0 02                              C305:  MOV     AL,2               ; SET BORDER REG
E55E  EE                                        OUT     DX,AL              ; SELECT VGA BORDER REG
E55F  8A C3                                     MOV     AL,BL              ; GET COLOR
E561  EE                                        OUT     DX,AL              ; SET IT
E562  A2 0066 R                                 MOV     CRT_PALLETTE,AL    ; SAVE THE COLOR VALUE
E565  E9 0F70 R                                 JMP     VIDEO_RETURN
                                         ;------ HANDLE COLOR 1 BY CHANGING PALETTE REGISTERS
E568  A0 0049 R                          C31:   MOV     AL,CRT_MODE        ; GET CURRENT MODE
E56B  B9 0D95 R                                 MOV     CX,OFFSET M0072    ; POINT TO 2 COLOR TABLE ENTRY
E56E  3C 06                                     CMP     AL,6               ; 2 COLOR MODE?
E570  74 0F                                     JE      C33                ; YES, JUMP
E572  3C 04                                     CMP     AL,4               ; 4 COLOR MODE?
E574  74 08                                     JE      C32                ; YES, JUMP
E576  3C 05                                     CMP     AL,5               ; 4 COLOR MODE?
E578  74 04                                     JE      C32                ; YES, JUMP
E57A  3C 0A                                     CMP     AL,0AH             ; 4 COLOR MODE?
E57C  75 20                                     JNE     C36                ; NO, GO TO 16 COLOR SET UP
E57E  B9 0D9D R                          C32:   MOV     CX,OFFSET M0074    ; POINT TO 4 COLOR TABLE ENTRY
E581  D0 CB                              C33:   ROR     BL,1               ; SELECT ALTERNATE SET?
E583  73 03                                     JNC     C34                ; NO, JUMP
E585  83 C1 04                                  ADD     CX,M0072L          ; POINT TO NEXT ENTRY
E588  8B D9                              C34:   MOV     BX,CX              ; TABLE ADDRESS IN BX
E58A  43                                        INC     BX                 ; SKIP OVER BACKGROUND COLOR
E58B  B9 0003                                   MOV     CX,M0072L-1        ; SET NUMBER OF REGS TO FILL
E58E  B4 11                                     MOV     AH,11H             ; AH IS REGISTER COUNTER
E590  8A C4                              C35:   MOV     AL,AH              ; GET REG NUMBER
E592  EE                                        OUT     DX,AL              ; SELECT IT
E593  2E: 8A 07                                 MOV     AL,CS:[BX]         ; GET DATA
E596  EE                                        OUT     DX,AL              ; SET IT
E597  FE C4                                     INC     AH                 ; NEXT REG
E599  43                                        INC     BX                 ; NEXT TABLE VALUE
E59A  E2 F4                                     LOOP    C35
E59C  EB 0D                                     JMP     SHORT C38
E59E  B4 11                              C36:   MOV     AH,11H             ; AH IS REGISTER COUNTER
E5A0  B9 000F                                   MOV     CX,15              ; NUMBER OF PALETTES
E5A3  8A C4                              C37:   MOV     AL,AH              ; GET REG NUMBER
E5A5  EE                                        OUT     DX,AL              ; SELECT IT
E5A6  EE                                        OUT     DX,AL              ; SET PALETTE VALUE
E5A7  FE C4                                     INC     AH                 ; NEXT REG
E5A9  E2 F8                                     LOOP    C37
E5AB  32 C0                              C38:   XOR     AL,AL              ; SELECT LOW REG TO ENABLE VIDEO
                                                                           ; AGAIN
E5AD  EE                                        OUT     DX,AL
E5AE  E9 0F70 R                                 JMP     VIDEO_RETURN
E5B1                                     SET_COLOR      ENDP
                                         ;----------------------------------------------
                                         ; VIDEO STATE
                                         ;  RETURNS THE CURRENT VIDEO STATE IN AX
                                         ;  AH = NUMBER OF COLUMNS ON THE SCREEN
                                         ;  AL = CURRENT VIDEO MODE
                                         ;  BH = CURRENT ACTIVE PAGE
                                         ;----------------------------------------------
E5B1                                     VIDEO_STATE    PROC    NEAR
E5B1  8A 26 004A R                              MOV     AH,BYTE PTR CRT_COLS ; GET NUMBER OF COLUMNS
E5B5  A0 0049 R                                 MOV     AL,CRT_MODE        ; CURRENT MODE
E5B8  8A 3E 0062 R                              MOV     BH,ACTIVE_PAGE     ; GET CURRENT ACTIVE PAGE
E5BC  5F                                        POP     DI                 ; RECOVER REGISTERS
E5BD  5E                                        POP     SI                 ;
E5BE  59                                        POP     CX                 ; DISCARD SAVED BX
E5BF  E9 0F73 R                                 JMP     C22                ; RETURN TO CALLER
E5C2                                     VIDEO_STATE    ENDP
                                         ;---------------------------------
                                         ; POSITION
                                         ;      THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
                                         ;      OF A CHARACTER IN THE ALPHA MODE
                                         ; INPUT
                                         ;      AX = ROW, COLUMN POSITION
                                         ; OUTPUT
                                         ;      AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
                                         ;---------------------------------
E5C2                                     POSITION       PROC    NEAR
E5C2  53                                        PUSH    BX                 ; SAVE REGISTER
E5C3  8B D8                                     MOV     BX,AX
E5C5  8A C4                                     MOV     AL,AH              ; ROWS TO AL
E5C7  F6 26 004A R                              MUL     BYTE PTR CRT_COLS       ; DETERMINE BYTES TO ROW
E5CB  32 FF                                     XOR     BH,BH
E5CD  03 C3                                     ADD     AX,BX              ; ADD IN COLUMN VALUE
E5CF  D1 E0                                     SAL     AX,1               ; * 2 FOR ATTRIBUTE BYTES
E5D1  5B                                        POP     BX
E5D2  C3                                        RET
E5D3                                     POSITION       ENDP
                                         ;--------------------------------------
                                         ; SCROLL UP
                                         ;      THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
                                         ;      ON THE SCREEN
                                         ; INPUT
                                         ;      (AH) = CURRENT CRT MODE
                                         ;      (AL) = NUMBER OF ROWS TO SCROLL
                                         ;      (CX) = ROW/COLUMN OF UPPER LEFT CORNER
                                         ;      (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
                                         ;      (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
                                         ;      (DS) = DATA SEGMENT
                                         ;      (ES) = REGEN BUFFER SEGMENT
                                         ; OUTPUT
                                         ;      NONE -- THE REGEN BUFFER IS MODIFIED
                                         ;--------------------------------------
```

```
                                        ASSUME  CS:CODE,DS:DATA,ES:DATA
E5D3                            SCROLL_UP       PROC    NEAR
E5D3  8A D8                             MOV     BL,AL           ; SAVE LINE COUNT IN BL
E5D5  80 FC 04                          CMP     AH,4            ; TEST FOR GRAPHICS MODE
E5D8  72 03                             JC      C39             ; HANDLE SEPARATELY
E5DA  E9 F259 R                         JMP     GRAPHICS_UP
E5DD                            C39:                            ; UP_CONTINUE
E5DD  53                                PUSH    BX              ; SAVE FILL ATTRIBUTE IN BH
E5DE  8B C1                             MOV     AX,CX           ; UPPER LEFT POSITION
E5E0  E8 E609 R                         CALL    SCROLL_POSITION ; DO SETUP FOR SCROLL
E5E3  74 20                             JZ      C44             ; BLANK_FIELD
E5E5  03 F0                             ADD     SI,AX           ; FROM ADDRESS
E5E7  8A E6                             MOV     AH,DH           ; # ROWS IN BLOCK
E5E9  2A E3                             SUB     AH,BL           ; # ROWS TO BE MOVED
E5EB  E8 E62F R                 C40:    CALL    C45             ; MOVE ONE ROW
E5EE  03 F5                             ADD     SI,BP
E5F0  03 FD                             ADD     DI,BP           ; POINT TO NEXT LINE IN BLOCK
E5F2  FE CC                             DEC     AH              ; COUNT OF LINES TO MOVE
E5F4  75 F5                             JNZ     C40             ; ROW_LOOP
E5F6  58                        C41:    POP     AX              ; RECOVER ATTRIBUTE IN AH
E5F7  B0 20                             MOV     AL,' '          ; FILL WITH BLANKS
E5F9  E8 E638 R                 C42:    CALL    C46             ; CLEAR THE ROW
E5FC  03 FD                             ADD     DI,BP           ; POINT TO NEXT LINE
E5FE  FE CB                             DEC     BL              ; COUNTER OF LINES TO SCROLL
E600  75 F7                             JNZ     C42             ; CLEAR_LOOP
E602  E9 0F70 R                 C43:    JMP     VIDEO_RETURN
E605  8A DE                     C44:    MOV     BL,DH           ; GET ROW COUNT
E607  EB ED                             JMP     C41             ; GO CLEAR THAT AREA
E609                            SCROLL_UP       ENDP
                                ;----- HANDLE COMMON SCROLL SET UP HERE
E609                            SCROLL_POSITION PROC    NEAR
E609  E8 E5C2 R                         CALL    POSITION        ; CONVERT TO REGEN POINTER
E60C  03 06 004E R                      ADD     AX,CRT_START    ; OFFSET OF ACTIVE PAGE
E610  8B F8                             MOV     DI,AX           ; TO ADDRESS FOR SCROLL
E612  8B F0                             MOV     SI,AX           ; FROM ADDRESS FOR SCROLL
E614  2B D1                             SUB     DX,CX           ; DX = #ROWS, #COLS IN BLOCK
E616  FE C6                             INC     DH
E618  FE C2                             INC     DL              ; INCREMENT FOR 0 ORIGIN
E61A  32 ED                             XOR     CH,CH           ; SET HIGH BYTE OF COUNT TO ZERO
E61C  8B 2E 004A R                      MOV     BP,CRT_COLS     ; GET NUMBER OF COLUMNS IN DISPLAY
E620  03 ED                             ADD     BP,BP           ; TIMES 2 FOR ATTRIBUTE BYTE
E622  8A C3                             MOV     AL,BL           ; GET LINE COUNT
E624  F6 26 004A R                      MUL     BYTE PTR CRT_COLS ; DETERMINE OFFSET TO FROM
                                                                ; ADDRESS
E628  03 C0                             ADD     AX,AX           ; *2 FOR ATTRIBUTE BYTE
E62A  06                                PUSH    ES              ; ESTABLISH ADDRESSING TO REGEN
                                                                ; BUFFER
E62B  1F                                POP     DS              ;  FOR BOTH POINTERS
E62C  0A DB                             OR      BL,BL           ; 0 SCROLL MEANS BLANK FIELD
E62E  C3                                RET                     ; RETURN WITH FLAGS SET
E62F                            SCROLL_POSITION ENDP
                                ;------ MOVE_ROW
E62F                            C45     PROC    NEAR
E62F  8A CA                             MOV     CL,DL           ; GET # OF COLS TO MOVE
E631  56                                PUSH    SI
E632  57                                PUSH    DI              ; SAVE START ADDRESS
E633  F3/ A5                            REP     MOVSW           ; MOVE THAT LINE ON SCREEN
E635  5F                                POP     DI
E636  5E                                POP     SI              ; RECOVER ADDRESSES
E637  C3                                RET
E638                            C45     ENDP
                                ;------ CLEAR_ROW
E638                            C46     PROC    NEAR
E638  8A CA                             MOV     CL,DL           ; GET # COLUMNS TO CLEAR
E63A  57                                PUSH    DI
E63B  F3/ AB                            REP     STOSW           ; STORE THE FILL CHARACTER
E63D  5F                                POP     DI
E63E  C3                                RET
E63F                            C46     ENDP
                                ;------------------------------------
                                ; SCROLL_DOWN
                                ;       THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
                                ;       BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
                                ;       WITH A DEFINED CHARACTER
                                ; INPUT
                                ;       (AH) = CURRENT CRT MODE
                                ;       (AL) = NUMBER OF LINES TO SCROLL
                                ;       (CX) = UPPER LEFT CORNER OF REGION
                                ;       (DX) = LOWER RIGHT CORNER OF REGION
                                ;       (BH) = FILL CHARACTER
                                ;       (DS) = DATA SEGMENT
                                ;       (ES) = REGEN SEGMENT
                                ; OUPUT
                                ;       NONE -- SCREEN IS SCROLLED
                                ;------------------------------------
E63F                            SCROLL_DOWN     PROC    NEAR
E63F  FD                                STD                     ; DIRECTION FOR SCROLL DOWN
E640  8A D8                             MOV     BL,AL           ; LINE COUNT TO BL
E642  80 FC 04                          CMP     AH,4            ; TEST FOR GRAPHICS
E645  72 03                             JC      C47
E647  E9 F305 R                         JMP     GRAPHICS_DOWN
E64A  53                        C47:    PUSH    BX              ; SAVE ATTRIBUTE IN BH
E64B  8B C2                             MOV     AX,DX           ; LOWER RIGHT CORNER
E64D  E8 E609 R                         CALL    SCROLL_POSITION ; GET REGEN LOCATION
E650  74 1F                             JZ      C51
E652  2B F0                             SUB     SI,AX           ; SI IS FROM ADDRESS
E654  8A E6                             MOV     AH,DH           ; GET TOTAL # ROWS
E656  2A E3                             SUB     AH,BL           ; COUNT TO MOVE IN SCROLL
```

```
E658  E8 E62F R              C48:    CALL    C45             ; MOVE ONE ROW
E65B  2B F5                          SUB     SI,BP
E65D  2B FD                          SUB     DI,BP
E65F  FE CC                          DEC     AH
E661  75 F5                          JNZ     C48
E663  58                     C49:    POP     AX              ; RECOVER ATTRIBUTE IN AH
E664  B0 20                          MOV     AL,' '
E666  E8 E638 R              C50:    CALL    C46             ; CLEAR ONE ROW
E669  2B FD                          SUB     DI,BP           ; GO TO NEXT ROW
E66B  FE CB                          DEC     BL
E66D  75 F7                          JNZ     C50
E66F  EB 91                          JMP     C43             ; SCROLL_END
E671  8A DE                  C51:    MOV     BL,DH
E673  EB EE                          JMP     C49
E675                         SCROLL_DOWN     ENDP
                             ;---------------------------------------
                             ; MODE_ALIVE
                             ;       THIS ROUTINE READS 256 LOCATIONS IN MEMORY AS EVERY OTHER
                             ;       LOCATION IN 512 LOCATIONS.  THIS IS TO INSURE THE DATA
                             ;       INTEGRITY OF MEMORY DURING MODE CHANGES.
                             ;---------------------------------------
E675                         MODE_ALIVE      PROC    NEAR
E675  50                             PUSH    AX             ;SAVE USED REGS
E676  56                             PUSH    SI
E677  51                             PUSH    CX
E678  33 F6                          XOR     SI,SI
E67A  B9 0100                        MOV     CX,256
E67D  AC                     C52:    LODSB
E67E  46                             INC     SI
E67F  E2 FC                          LOOP    C52
E681  59                             POP     CX
E682  5E                             POP     SI
E683  58                             POP     AX
E684  C3                             RET
E685                         MODE_ALIVE      ENDP
                             ;--------------------------------------------------------
                             ; SET_PALLETTE
                             ;       THIS ROUTINE WRITES THE PALETTE REGISTERS
                             ; INPUT
                             ;       (AL) = 0        SET PALETTE REG
                             ;               (BH) =  VALUE TO SET
                             ;               (BL) =  PALETTE REG TO SET
                             ;       (AL) = 1        SET BORDER COLOR REG
                             ;               (BH) =  VALUE TO SET
                             ;       (AL) = 2        SET ALL PALETTE REGS AND BORDER REG
                             ;       NOTE: REGISTERS ARE WRITE ONLY.
                             ;--------------------------------------------------------
E685                         SET_PALLETTE    PROC    NEAR
E685  50                             PUSH    AX
E686  8B F4                          MOV     SI,SP
E688  36: 8B 44 0C                   MOV     AX,SS:[SI+12]   ; GET SEG FROM STACK
E68C  8E C0                          MOV     ES,AX
E68E  8B F2                          MOV     SI,DX           ; OFFSET IN SI
E690  BA 03DA                        MOV     DX,VGA_CTL      ; SET VGA CONTROL PORT
E693  EC                     C53:    IN      AL,DX           ; GET VGA STATUS
E694  24 08                          AND     AL,08H          ; IN VERTICAL RETRACE?
E696  75 FB                          JNZ     C53             ; YES, WAIT FOR IT TO GO AWAY
E698  EC                     C54:    IN      AL,DX           ; GET VGA STATUS
E699  24 08                          AND     AL,08H          ; IN VERITCAL RETRACE?
E69B  74 FB                          JZ      C54             ; NO, WAIT FOR IT
E69D  58                             POP     AX
E69E  0A C0                          OR      AL,AL           ; SET PALETTE REG?
E6A0  74 0C                          JZ      C55             ; YES, GO DO IT
E6A2  3C 02                          CMP     AL,2            ; SET ALL REGS?
E6A4  74 17                          JE      C57             ;
E6A6  3C 01                          CMP     AL,1            ; SET BORDER COLOR REG?
E6A8  75 2B                          JNE     C59             ; NO, DON'T DO ANYTHING
E6AA  B0 02                          MOV     AL,2            ; SET BORDER COLOR REG NUMBER
E6AC  EB 06                          JMP     SHORT C56
E6AE  8A C3                  C55:    MOV     AL,BL           ; GET DESIRED REG NUMBER IN AL
E6B0  24 0F                          AND     AL,0FH          ; STRIP UNUSED BITS
E6B2  0C 10                          OR      AL,10H          ; MAKE INTO REAL REG NUMBER
E6B4  EE                     C56:    OUT     DX,AL           ; SELECT REG
E6B5  8A C7                          MOV     AL,BH           ; GET DATA IN AL
E6B7  EE                             OUT     DX,AL           ; SET NEW DATA
E6B8  32 C0                          XOR     AL,AL           ; SET REG 0 SO DISPLAY WORKS AGAIN
E6BA  EE                             OUT     DX,AL
E6BB  EB 18                          JMP     SHORT C59
E6BD  B4 10                  C57:    MOV     AH,10H          ; AH IS REG COUNTER
E6BF  8A C4                  C58:    MOV     AL,AH           ; REG ADDRESS IN AL
E6C1  EE                             OUT     DX,AL           ; SELECT IT
E6C2  26: 8A 04                      MOV     AL,BYTE PTR ES:[SI]      ;GET DATA
E6C5  EE                             OUT     DX,AL           ; PUT IN VGA REG
E6C6  46                             INC     SI              ; NEXT DATA BYTE
E6C7  FE C4                          INC     AH              ; NEXT REG
E6C9  80 FC 20                       CMP     AH,20H          ; LAST PALETTE REG?
E6CC  72 F1                          JB      C58             ; NO, DO NEXT ONE
E6CE  B0 02                          MOV     AL,2            ; SET BORDER REG
E6D0  EE                             OUT     DX,AL           ; SELECT IT
E6D1  26: 8A 04                      MOV     AL,BYTE PTR ES:[SI]      ; GET DATA
E6D4  EE                             OUT     DX,AL           ; PUT IN VGA REG
```

```
E6D4  EE                        OUT     DX,AL           ; PUT IN VGA REG
E6D5  E9 0F70 R         C59:    JMP     VIDEO_RETURN    ; ALL DONE
E6D8                    SET_PALLETTE    ENDP
E6D8                    MFG_UP  PROC    NEAR
E6D8  50                        PUSH    AX
E6D9  1E                        PUSH    DS
                                ASSUME  DS:XXDATA
E6DA  B8 ---- R                 MOV     AX,XXDATA
E6DD  8E D8                     MOV     DS,AX
E6DF  A0 0005 R                 MOV     AL,MFG_TST      ; GET MFG CHECKPOINT
E6E2  E6 10                     OUT     10H,AL          ; OUTPUT IT TO TESTER
E6E4  FE C8                     DEC     AL              ; DROP IT BY 1 FOR THE NEXT TEST
E6E6  A2 0005 R                 MOV     MFG_TST,AL
                                ASSUME  DS:ABS0
E6E9  1F                        POP     DS
E6EA  58                        POP     AX
E6EB  C3                        RET
E6EC                    MFG_UP  ENDP
                                ASSUME  CS:CODE,DS:DATA
E6F2                            ORG     0E6F2H
E6F2  E9 0B1B R                 JMP     NEAR PTR BOOT_STRAP
                        ;------------------------------------------------------------------
                        ;   SUBROUTINE TO SET UP CONDITIONS FOR THE TESTING OF 8250 AND
                        ;       8259 INTERRUPTS.  ENABLES MASKABLE EXTERNAL INTERRUPTS,
                        ;       CLEARS THE 8259 INTR RECEIVED FLAG BIT, AND ENABLES THE
                        ;       DEVICE'S 8259 INTR (WHICHEVER IS BEING TESTED).
                        ;   IT EXPECTS TO BE PASSED:
                        ;       (DS) = ADDRESS OF SEGMENT WHERE INTR_FLAG IS DEFINED
                        ;       (DI) = OFFSET OF THE INTERRUPT BIT MASK
                        ;   UPON RETURN:
                        ;       INTR_FLAG BIT FOR THE DEVICE = 0
                        ;   NO REGISTERS ARE ALTERED.
                        ;------------------------------------------------------------------
E6F5                    SUI     PROC    NEAR
E6F5  50                        PUSH    AX
E6F6  FB                        STI                     ; ENABLE MASKABLE EXTERNAL
                                                        ; INTERRUPTS
E6F7  2E: 8A 25                 MOV     AH,CS:[DI]      ; GET INTERRUPT BIT MASK
E6FA  20 26 0084 R              AND     INTR_FLAG,AH    ; CLEAR 8259 INTERRUPT REC'D FLAG
                                                        ; BIT
E6FE  E4 21                     IN      AL,INTA01       ; CURRENT INTERRUPTS
E700  22 C4                     AND     AL,AH           ; ENABLE THIS INTERRUPT, TOO
E702  E6 21                     OUT     INTA01,AL       ; WRITE TO 8259 (INTERRUPT
                                                        ; CONTROLLER)
E704  58                        POP     AX
E705  C3                        RET
E706                    SUI     ENDP
                        ;------------------------------------------------------------------
                        ;   SUBROUTINE WHICH CHECKS IF A 8259 INTERRUPT IS GENERATED BY THE
                        ;       8250 INTERRUPT.
                        ;   IT EXPECTS TO BE PASSED:
                        ;       (DI) = OFFSET OF INTERRUPT BIT MASK
                        ;       (DS) = ADDRESS OF SEGMENT WHERE INTR_FLAG IS DEFINED.
                        ;   IT RETURNS:
                        ;       (CF) = 1 IF NO INTERRUPT IS GENERATED
                        ;              0 IF THE INTERRUPT OCCURRED
                        ;       (AL) = COMPLEMENT OF THE INTERRUPT MASK
                        ;   NO OTHER REGISTERS ARE ALTERED.
                        ;------------------------------------------------------------------
E706                    C5059   PROC    NEAR
E706  51                        PUSH    CX
E707  2B C9                     SUB     CX,CX           ; SET PROGRAM LOOP COUNT
E709  2E: 8A 05                 MOV     AL,CS:[DI]      ; GET INTERRUPT MASK
E70C  34 FF                     XOR     AL,0FFH         ; COMPLEMENT MASK SO ONLY THE INTR
                                                        ; TEST BIT IS ON
E70E  84 06 0084 R      AT25:   TEST    INTR_FLAG,AL    ; 8259 INTERRUPT OCCUR?
E712  75 03                     JNE     AT27            ; YES - CONTINUE
E714  E2 F8                     LOOP    AT25            ; WAIT SOME MORE
E716  F9                        STC                     ; TIME'S UP - FAILED
E717  59                AT27:   POP     CX
E718  C3                        RET
E719                    C5059   ENDP
                        ;------------------------------------------------------------------
                        ;   SUBROUTINE TO WAIT FOR ALL ENABLED 8250 INTERRUPTS TO CLEAR (SO
                        ;       NO INTRS WILL BE PENDING).  EACH INTERRUPT COULD TAKE UP TO
                        ;       1 MILLISECOND TO CLEAR. THE INTERRUPT IDENTIFICATION
                        ;       REGISTER WILL BE CHECKED UNTIL THE INTERRUPT(S) IS CLEARED
                        ;       OR A TIMEOUT OCCURS.
                        ;   EXPECTS TO BE PASSED:
                        ;       (DX) = ADDRESS OF THE INTERRUPT ID REGISTER
                        ;   RETURNS:
                        ;       (AL) = CONTENTS OF THE INTR ID REGISTER
                        ;       (CF) = 1  IF INTERRUPTS ARE STILL PENDING
                        ;              0  IF NO INTERRUPTS ARE PENDING (ALL CLEAR)
                        ;   NO OTHER REGISTERS ARE ALTERED.
                        ;------------------------------------------------------------------
E719                    W8250C  PROC    NEAR
E719  51                        PUSH    CX
E71A  2B C9                     SUB     CX,CX
E71C  EC                AT28:   IN      AL,DX           ; READ INTR ID REG
E71D  3C 01                     CMP     AL,1            ; INTERRUPTS STILL PENDING?
E71F  74 05                     JE      AT29            ; NO - GOOD FINISH
E721  E2 F9                     LOOP    AT28            ; KEEP TRYING
E723  F9                        STC                     ; TIME'S UP - ERROR
E724  EB 01                     JMP     SHORT AT30
E726  F8                AT29:   CLC
E727  59                AT30:   POP     CX
E728  C3                        RET
E729                    W8250C  ENDP
```

```
                                        ;-----INT 14-------------------------------------------
                                        ;RS232_IO
                                        ;       THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
                                        ;       PORT ACCORDING TO THE PARAMETERS:
                                        ;       (AH)=0  INITIALIZE THE COMMUNICATIONS PORT
                                        ;               (AL) HAS PARMS FOR INITIALIZATION
                                        ;
                                        ;---7-------6-------5-------4-------3-------2---------1-------0----
                                        ;-------- BAUD RATE ---::----PARITY----::-STOPBIT-::--WORD LENGTH--
                                        ;
                                        ;  000 - 110              X0 - NONE       0 - 1     10 - 7 BITS
                                        ;  001 - 150              01 - ODD        1 - 2     11 - 8 BITS
                                        ;  010 - 300              11 - EVEN
                                        ;  011 - 600
                                        ;  100 - 1200
                                        ;  101 - 2400
                                        ;  110 - 4800
                                        ;  111 - 4800
                                        ;               ON RETURN, THE RS232 INTERRUPTS ARE DISABLED AND
                                        ;                          CONDITIONS ARE SET AS IN CALL TO COMMO
                                        ;                          STATUS (AH=3)
                                        ;       (AH)=1  SEND THE CHARACTER IN (AL) OVER THE COMMO LINE
                                        ;               (AL) REGISTER IS PRESERVED
                                        ;               ON EXIT, BIT 7 OF AH IS SET IF THE ROUTINE WAS
                                        ;                       UNABLE TO TRANSMIT THE BYTE OF DATA OVER
                                        ;                       THE LINE. IF BIT 7 OF AH IS NOT SET, THE
                                        ;                       REMAINDER OF AH IS SET AS IN A STATUS
                                        ;                       REQUEST, REFELECTING THE CURRENT STATUS OF
                                        ;                       THE LINE.
                                        ;       (AH)=2  RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
                                        ;                       RETURNING TO CALLER
                                        ;               ON EXIT, AH HAS THE CURRENT LINE STATUS, AS SET BY
                                        ;                       THE STATUS ROUTINE, EXCEPT THAT THE ONLY
                                        ;                       BITS LEFT ON, ARE THE ERROR BITS
                                        ;                       (7,4,3,2,1). IN THIS CASE, THE TIME OUT BIT
                                        ;                       INDICATES DATA SET READY WAS NOT RECEIVED.
                                        ;                       THUS, AH IS NON ZERO ONLY WHEN AN ERROR
                                        ;                       OCCURRED.(NOTE: IF THE TIME-OUT BIT IS SET,
                                        ;                       OTHER BITS IN AH MAY NOT BE RELIABLE.)
                                        ;       (AH)=3  RETURN THE COMMO PORT STATUS IN (AX)
                                        ;               AH CONTAINS THE LINE CONTROL STATUS
                                        ;               BIT 7 = TIME OUT
                                        ;               BIT 6 = TRANS SHIFT REGISTER EMPTY
                                        ;               BIT 5 = TRAN HOLDING REGISTER EMPTY
                                        ;               BIT 4 = BREAK DETECT
                                        ;               BIT 3 = FRAMING ERROR
                                        ;               BIT 2 = PARITY ERROR
                                        ;               BIT 1 = OVERRUN ERROR
                                        ;               BIT 0 = DATA READY
                                        ;               AL CONTAINS THE MODEM STATUS
                                        ;               BIT 7 = RECIEVED LINE SIGNAL DETECT
                                        ;               BIT 6 = RING INDICATOR
                                        ;               BIT 5 = DATA SET READY
                                        ;               BIT 4 = CLEAR TO SEND
                                        ;               BIT 3 = DELTA RECEIVE LINE SIGNAL DETECT
                                        ;               BIT 2 = TRAILING EDGE RING DETECTOR
                                        ;               BIT 1 = DELTA DATA SET READY
                                        ;               BIT 0 = DELTA CLEAR TO SEND
                                        ;       (DX) = PARAMETER INDICATING WHICH RS232 CARD (0,1 ALLOWED)
                                        ; DATA AREA RS232_BASE CONTAINS THE BASE ADDRESS OF THE 8250 ON THE
                                        ;    CARD.  LOCATION 400H CONTAINS UP TO 4 RS232 ADDRESSES POSSIBLE
                                        ; DATA AREA RS232_TIM_OUT (BYTE) CONTAINS OUTER LOOP COUNT
                                        ;       VALUE FOR TIMEOUT (DEFAULT=1)
                                        ;OUTPUT
                                        ;               AX      MODIFIED ACCORDING TO PARMS OF CALL
                                        ;               ALL OTHERS UNCHANGED
                                        ;------------------------------------------------
                                                ASSUME  CS:CODE,DS:DATA
E729                                            ORG     0E729H
E729                                    A1      LABEL   WORD
E729  03F9                                      DW      1017    ; 110 BAUD      ; TABLE OF INIT VALUE
E72B  02EA                                      DW      746     ; 150
E72D  0175                                      DW      373     ; 300
E72F  00BA                                      DW      186     ; 600
E731  005D                                      DW      93      ; 1200
E733  002F                                      DW      47      ; 2400
E735  0017                                      DW      23      ; 4800
E737  0017                                      DW      23      ; 4800
E739                                    RS232_IO        PROC    FAR
                                        ;------ VECTOR TO APPROPRIATE ROUTINE
E739  FB                                        STI                     ; INTERRUPTS BACK ON
E73A  1E                                        PUSH    DS              ; SAVE SEGMENT
E73B  52                                        PUSH    DX
E73C  56                                        PUSH    SI
E73D  57                                        PUSH    DI
E73E  51                                        PUSH    CX
E73F  53                                        PUSH    BX
E740  8B F2                                     MOV     SI,DX           ; RS232 VALUE TO SI
E742  8B FA                                     MOV     DI,DX           ; AND TO DI (FOR TIMEOUTS)
E744  D1 E6                                     SHL     SI,1            ; WORD OFFSET
E746  E8 138B R                                 CALL    DDS             ; POINT TO BIOS DATA SEGMENT
E749  8B 94 0000 R                              MOV     DX,RS232_BASE[SI] ; GET BASE ADDRESS
E74D  0B D2                                     OR      DX,DX           ; TEST FOR 0 BASE ADDRESS
E74F  74 13                                     JZ      A3              ; RETURN
E751  0A E4                                     OR      AH,AH           ; TEST FOR (AH)=0
E753  74 16                                     JZ      A4              ; COMMUN INIT
E755  FE CC                                     DEC     AH              ; TEST FOR (AH)=1
E757  74 47                                     JZ      A5              ; SEND AL
E759  FE CC                                     DEC     AH              ; TEST FOR (AH)=2
E75B  74 6C                                     JZ      A12             ; RECEIVE INTO AL
E75D  FE CC                                     DEC     AH              ; TEST FOR (AH)=3
E75F  75 03                                     JNZ     A3
E761  E9 E7F3 R                                 JMP     A18             ; COMMUNICATION STATUS
```

```
E764                           A3:                             ; RETURN FROM RS232
E764  5B                               POP    BX
E765  59                               POP    CX
E766  5F                               POP    DI
E767  5E                               POP    SI
E768  5A                               POP    DX
E769  1F                               POP    DS
E76A  CF                               IRET                    ; RETURN TO CALLER, NO ACTION
                                       ;------ INITIALIZE THE COMMUNICATIONS PORT
E76B  8A E0                    A4:     MOV    AH,AL            ; SAVE INIT PARMS IN AH
E76D  83 C2 03                         ADD    DX,3             ; POINT TO 8250 CONTROL REGISTER
E770  B0 80                            MOV    AL,80H
E772  EE                               OUT    DX,AL            ; SET DLAB=1
                               ;------ DETERMINE BAUD RATE DIVISOR
E773  8A D4                            MOV    DL,AH            ; GET PARMS TO DL
E775  B1 04                            MOV    CL,4
E777  D2 C2                            ROL    DL,CL
E779  81 E2 000E                       AND    DX,0EH           ; ISOLATE THEM
E77D  BF E729 R                        MOV    DI,OFFSET A1     ; BASE OF TABLE
E780  03 FA                            ADD    DI,DX            ; PUT INTO INDEX REGISTER
E782  8B 94 0000 R                     MOV    DX,RS232_BASE[SI] ; POINT TO HIGH ORDER OF DIVISOR
E786  42                               INC    DX
E787  2E: 8A 45 01                     MOV    AL,CS:[DI]+1     ; GET HIGH ORDER OF DIVISOR
E78B  EE                               OUT    DX,AL            ; SET MS OF DIV TO 0
E78C  4A                               DEC    DX
E78D  2E: 8A 05                        MOV    AL,CS:[DI]       ; GET LOW ORDER OF DIVISOR
E790  EE                               OUT    DX,AL            ; SET LOW OF DIVISOR
E791  83 C2 03                         ADD    DX,3
E794  8A C4                            MOV    AL,AH            ; GET PARMS BACK
E796  24 1F                            AND    AL,01FH          ; STRIP OFF THE BAUD BITS
E798  EE                               OUT    DX,AL            ; LINE CONTROL TO 8 BITS
E799  4A                               DEC    DX
E79A  4A                               DEC    DX
E79B  B0 00                            MOV    AL,0
E79D  EE                               OUT    DX,AL            ; INTERRUPT ENABLES ALL OFF
E79E  EB 53                            JMP    SHORT A18        ; COM_STATUS
                               ;------ SEND CHARACTER IN (AL) OVER COMMO LINE
E7A0                           A5:
E7A0  50                               PUSH   AX               ; SAVE CHAR TO SEND
E7A1  83 C2 04                         ADD    DX,4             ; MODEM CONTROL REGISTER
E7A4  B0 03                            MOV    AL,3             ; DTR AND RTS
E7A6  EE                               OUT    DX,AL            ; DATA TERMINAL READY, REQUEST TO
                                                               ; SEND
E7A7  42                               INC    DX               ; MODEM STATUS REGISTER
E7A8  42                               INC    DX
E7A9  B7 30                            MOV    BH,30H           ; DATA SET READY & CLEAR TO SEND
E7AB  E8 E802 R                        CALL   WAIT_FOR_STATUS  ; ARE BOTH TRUE?
E7AE  74 08                            JE     A9               ; YES, READY TO TRANSMIT CHAR
E7B0  59                       A7:     POP    CX
E7B1  8A C1                            MOV    AL,CL            ; RELOAD DATA BYTE
E7B3  80 CC 80                 A8:     OR     AH,80H           ; INDICATE TIME OUT
E7B6  EB AC                            JMP    A3               ; RETURN
E7B8                           A9:                             ; CLEAR_TO_SEND
E7B8  4A                               DEC    DX               ; LINE STATUS REGISTER
E7B9  B7 20                            MOV    BH,20H           ; IS TRANSMITTER READY
E7BB  E8 E802 R                        CALL   WAIT_FOR_STATUS  ; TEST FOR TRANSMITTER READY
E7BE  75 F0                            JNZ    A7               ; RETURN WITH TIME OUT SET
E7C0  83 EA 05                         SUB    DX,5             ; DATA PORT
E7C3  59                               POP    CX               ; RECOVER IN CX TEMPORARILY
E7C4  8A C1                            MOV    AL,CL            ; MOVE CHAR TO AL FOR OUT, STATUS
                                                               ; IN AH
E7C6  EE                               OUT    DX,AL            ; OUTPUT CHARACTER
E7C7  EB 9B                            JMP    A3               ; RETURN
                               ;------ RECEIVE CHARACTER FROM COMMO LINE
E7C9  83 C2 04                 A12:    ADD    DX,4             ; MODEM CONTROL REGISTER
E7CC  B0 01                            MOV    AL,1             ; DATA TERMINAL READY
E7CE  EE                               OUT    DX,AL
E7CF  42                               INC    DX               ; MODEM STATUS REGISTER
E7D0  42                               INC    DX
E7D1  B7 20                            MOV    BH,20H           ; DATA SET READY
E7D3  E8 E802 R                        CALL   WAIT_FOR_STATUS  ; TEST FOR DSR
E7D6  75 DB                            JNZ    A8               ; RETURN WITH ERROR
E7D8  4A                               DEC    DX               ; LINE STATUS REGISTER
E7D9  EC                       A16:    IN     AL,DX
E7DA  A8 01                            TEST   AL,1             ; RECEIVE BUFFER FULL
E7DC  75 09                            JNZ    A17              ; TEST FOR REC. BUFF. FULL
E7DE  F6 06 0071 R 80                  TEST   BIOS_BREAK,80H   ; TEST FOR BREAK KEY
E7E3  74 F4                            JZ     A16              ; LOOP IF NO BREAK KEY
E7E5  EB CC                            JMP    A8               ; SET TIME OUT ERROR
E7E7  24 1E                    A17:    AND    AL,00011110B     ; TEST FOR ERROR CONDITIONS ON RECV
                                                               ; CHAR
E7E9  8A E0                            MOV    AH,AL
E7EB  8B 94 0000 R                     MOV    DX,RS232_BASE[SI] ; DATA PORT
E7EF  EC                               IN     AL,DX            ; GET CHARACTER FROM LINE
E7F0  E9 E764 R                        JMP    A3               ; RETURN
                               ;------ COMMO PORT STATUS ROUTINE
E7F3  8B 94 0000 R             A18:    MOV    DX,RS232_BASE[SI]
E7F7  83 C2 05                         ADD    DX,5             ; CONTROL PORT
E7FA  EC                               IN     AL,DX            ; GET LINE CONTROL STATUS
E7FB  8A E0                            MOV    AH,AL            ; PUT IN AH FOR RETURN
E7FD  42                               INC    DX               ; POINT TO MODEM STATUS REGISTER
E7FE  EC                               IN     AL,DX            ; GET MODEM CONTROL STATUS
E7FF  E9 E764 R                        JMP    A3               ; RETURN
                               ;------------------------------------------
                               ;       WAIT FOR STATUS ROUTINE
                               ;ENTRY: BH=STATUS BIT(S) TO LOOK FOR,
                               ;       DX=ADDR. OF STATUS REG
                               ;EXIT:  ZERO FLAG ON = STATUS FOUND
                               ;       ZERO FLAG OFF = TIMEOUT.
                               ;       AH=LAST STATUS READ
                               ;------------------------------------------
```

```
E802                        WAIT_FOR_STATUS PROC    NEAR
E802  8A 9D 007C R                  MOV     BL,RS232_TIM_OUT[DI] ;LOAD OUTER LOOP COUNT
E806  2B C9                 WFS0:   SUB     CX,CX
E808  EC                    WFS1:   IN      AL,DX           ;GET STATUS
E809  8A E0                         MOV     AH,AL           ; MOVE TO AH
E80B  22 C7                         AND     AL,BH           ;ISOLATE BITS TO TEST
E80D  3A C7                         CMP     AL,BH           ;EXACTLY = TO MASK
E80F  74 08                         JE      WFS_END         ;RETURN WITH ZERO FLAG ON
E811  E2 F5                         LOOP    WFS1            ;TRY AGAIN
E813  FE CB                         DEC     BL
E815  75 EF                         JNZ     WFS0
E817  0A FF                         OR      BH,BH           ;SET ZERO FLAG OFF
E819                        WFS_END:
E819  C3                            RET
E81A                        WAIT_FOR_STATUS ENDP
E81A                        RS232_IO        ENDP
                            ;-----------------------------------------------------------------
                            ;THIS ROUTINE WILL READ TIMER1.  THE VALUE READ IS RETURNED IN AX.
                            ;-----------------------------------------------------------------
E81A                        READ_TIME PROC NEAR
E81A  B0 40                         MOV     AL,40H          ; LATCH TIMER1
E81C  E6 43                         OUT     TIM_CTL,AL
E81E  50                            PUSH    AX              ; WAIT FOR 8253 TO INIT ITSELF
E81F  58                            POP     AX
E820  E4 41                         IN      AL,TIMER+1      ; READ LSB
E822  8A E0                         MOV     AH,AL           ; SAVE IT IN HIGH BYTE
E824  50                            PUSH    AX              ; WAIT FOR 8253 TO INIT ITSELF
E825  58                            POP     AX
E826  E4 41                         IN      AL,TIMER+1      ; READ MSB
E828  86 C4                         XCHG    AL,AH           ; PUT BYTES IN PROPER ORDER
E82A  C3                            RET
E82B                        READ_TIME       ENDP
E82E                                ORG     0E82EH
E82E  E9 13DD R                     JMP     NEAR PTR KEYBOARD_IO
                            ;-----------------------------------------------------------------
                            ;ASYNCHRONOUS COMMUNICATIONS ADAPTER POWER ON DIAGNOSTIC TEST
                            ;DESCRIPTION:
                            ;  THIS SUBROUTINE PERFORMS A THOROUGH CHECK OUT OF AN INS8250 LSI
                            ;  CHIP.
                            ;  THE TEST INCLUDES:
                            ;  1) INITIALIZATION OF THE CHIP TO ASSUME ITS MASTER RESET STATE.
                            ;  2) READING REGISTERS FOR KNOWN PERMANENT ZERO BITS.
                            ;  3) TESTING THE INS8250 INTERRUPT SYSTEM AND THAT THE 8250
                            ;     INTERRUPTS TRIGGER AN 8259 (INTERRUPT CONTROLLER) INTERRUPT.
                            ;  4) PERFORMING THE LOOP BACK TEST:
                            ;     A) TESTING WHAT WAS WRITTEN/READ AND THAT THE TRANSMITTER
                            ;        HOLDING REG EMPTY BIT AND THE RECEIVER INTERRUPT WORK
                            ;        PROPERLY.
                            ;     B) TESTING IF CERTAIN BITS OF THE DATA SET CONTROL REGISTER
                            ;        ARE 'LOOPED BACK' TO THOSE IN THE DATA SET STATUS
                            ;        REGISTER.
                            ;     C) TESTING THAT THE TRANSMITTER IS IDLE WHEN TRANSMISSION
                            ;        TEST IS FINISHED.
                            ; THIS SUBROUTINE EXPECTS TO HAVE THE FOLLOWING PARAMETER PASSED:
                            ;   (DX)= ADDRESS OF THE INS8250 CARD TO TEST.
                            ;   NOTE: THE ASSUMPTION HAS BEEN MADE THAT THE MODEM ADAPTER IS
                            ;         ----  LOCATED AT 03F8H; THE SERIAL PRINTER AT 02F8H.
                            ; IT RETURNS:
                            ;  (CF) = 1 IF ANY PORTION OF THE TEST FAILED
                            ;       = 0 IF TEST PASSED
                            ;  (BX) = FAILURE KEY FOR ERROR MESSAGE (ONLY VALID IF TEST FAILED)
                            ;     (BH) = 23H  SERIAL PRINTER ADAPTER TEST FAILURE
                            ;          = 24H  MODEM ADAPTER TEST FAILURE
                            ;     (BL) = 2   PERMANENT ZERO BITS IN INTERRUPT ENABLE REGISTER
                            ;                WERE INCORRECT
                            ;            3   PERMANENT ZERO BITS IN INTERRUPT IDENTIFICATION
                            ;                REGISTER WERE INCORRECT
                            ;            4   PERMANENT ZERO BITS IN DATA SET CONTROL REGISTER
                            ;                WERE INCORRECT
                            ;            5   PERMANENT ZERO BITS IN THE LINE STATUS REGISTER
                            ;                WERE INCORRECT
                            ;            6   RECEIVED DATA AVAILABLE INTERRUPT TEST FAILED
                            ;                (THE INTERRUPT WAS NOT GENERATED)
                            ;            16H  RECEIVED DATA AVAILABLE INTERRUPT FAILED TO CLEAR
                            ;             7   RESERVED FOR REPORTING THE TRANSMITTER HOLDING
                            ;                 REGISTER EMPTY INTERRUPT TEST FAILED
                            ;                 (NOT USED AT THIS TIME BECAUSE OF THE DIFFERENCES
                            ;                 BETWEEN THE 8250'S WHICH WILL BE USED)
                            ;           17H  TRANSMITTER HOLDING REG EMPTY INTR FAILED TO CLEAR
                            ;            8-B  RECEIVER LINE STATUS INTERRUPT TEST FAILED
                            ;                 (THE INTERRUPT WAS NOT GENERATED)
                            ;                 8 - OVERRUN ERROR
                            ;                 9 - PARITY ERROR
                            ;                 A - FRAMING ERROR
                            ;                 B - BREAK INTERRUPT ERROR
                            ;           18-1B RECEIVER LINE STATUS INTERRUPT FAILED TO CLEAR
                            ;            C-F  MODEM STATUS INTERRUPT TEST FAILED
                            ;                 (THE INTERRUPT WAS NOT GENERATED)
                            ;                 C - DELTA CLEAR TO SEND ERROR
                            ;                 D - DELTA DATA SET READY ERROR
                            ;                 E - TRAILING EDGE RING INDICATOR ERROR
                            ;                 F - DELTA RECEIVE LINE SIGNAL DETECT ERROR
                            ;
```

```
                                ;           1C-1F MODEM STATUS INTERRUPT FAILED TO CLEAR
                                ;            10H  AN 8250 INTERRUPT OCCURRED AS EXPECTED, BUT NO
                                ;                 8259 (INTR CONTROLLER) INTERRUPT WAS GENERATED
                                ;            11H  DURING THE TRANSMISSION TEST, THE TRANSMITTER
                                ;                 HOLDING REGISTER WAS NOT EMPTY WHEN IT SHOULD
                                ;                 HAVE BEEN.
                                ;            12H  DURING THE TRANSMISSION TEST, THE RECEIVED DATA
                                ;                 AVAILABLE INTERRUPT DIDN'T OCCUR.
                                ;            13H  TRANSMISSION ERROR - THE CHARACTER RECEIVED
                                ;                 DURING LOOP MODE WAS NOT THE SAME AS THE ONE
                                ;                 TRANSMITTED
                                ;            14H  DURING TRANSMISSION TEST, THE 4 DATA SET CONTROL
                                ;                 OUTPUTS WERE NOT THE SAME AS THE 4 DATA SET
                                ;                 CONTROL INPUTS.
                                ;            15H  THE TRANSMITTER WAS NOT IDLE AFTER THE TRANS-
                                ;                 MISSION TEST COMPLETED.
                                ;   ON EXIT:
                                ;       - THE MODEM OR SERIAL PRINTER'S 8259 INTERRUPT (WHICHEVER
                                ;         DEVICE WAS TESTED) IS DISABLED.
                                ;       - THE 8250 IS IN THE MASTER RESET STATE.
                                ;   ONLY THE DS REGISTER IS PRESERVED - ALL OTHERS ARE ALTERED.
                                ;-----------------------------------------------------------------
= 0084                          WRAP    EQU     84H             ; LOOP BACK TRANSMISSION TEST
                                                                ; INTERRUPT VECTOR ADDRESS
                                                                ; (IN DIAGNOSTICS)
                                        ASSUME  CS:CODE,DS:DATA
E831                            UART    PROC       NEAR
E831  1E                                PUSH    DS
E832  E4 21                             IN      AL,INTA01       ; CURRENT ENABLED INTERRUPTS
E834  50                                PUSH    AX              ; SAVE FOR EXIT
E835  0C 01                             OR      AL,00000001B    ; DISABLE TIMER INTR DURING THIS
                                                                ; TEST
E837  E6 21                             OUT     INTA01,AL
E839  9C                                PUSHF                   ; SAVE CALLER'S FLAGS (SAVE INTR
                                                                ; FLAG)
E83A  52                                PUSH    DX              ; SAVE BASE ADDRESS OF ADAPTER CARD
E83B  E8 138B R                         CALL    DDS             ; SET UP 'DATA' AS DATA SEGMENT
                                                                ; ADDRESS
                                ;-----------------------------------------------------------------
                                ;       INITIALIZE PORTS FOR MASTER RESET STATES AND TEST PERMANENT
                                ;       ZERO DATA BITS FOR CERTAIN PORTS.
                                ;-----------------------------------------------------------------
E83E  E8 0AC4 R                         CALL    I8250
E841  73 03                             JNC     AT1             ; ALL OK
E843  E9 E94B R                         JMP     AT14            ; A PORT'S ZERO BITS WERE NOT ZERO!
                                ;-----------------------------------------------------------------
                                ;       INS8250 INTERRUPT SYSTEM TEST
                                ;       ONLY THE INTERRUPT BEING TESTED WILL BE ENABLED.
                                ;-----------------------------------------------------------------
                                ;       SET DI AND SI FOR CALLS TO 'SUI'
E846  BF 0041 R                 AT1:    MOV     DI,OFFSET IMASKS ; BASE ADDRESS OF INTERRUPT MASKS
E849  33 F6                             XOR     SI,SI           ; MODEM INDEX
E84B  80 FE 02                          CMP     DH,2            ; OR SERIAL?
E84E  75 02                             JNE     AT2             ; NO - IT'S MODEM
E850  46                                INC     SI              ; IT'S SERIAL PRINTER
E851  47                                INC     DI              ; SERIAL PRINTER 8259 MASK ADDRESS
                                ;       RECEIVED DATA AVAILABLE INTERRUPT TEST
E852  E8 E6F5 R                 AT2:    CALL    SUI             ; SET UP FOR INTERRUPTS
E855  FE C3                             INC     BL              ; ERROR REPORTER (INIT. IN I8250)
E857  42                                INC     DX              ; POINT TO INTERRUPT ENABLE
                                                                ; REGISTER
E858  B0 01                             MOV     AL,1            ; ENABLE RECEIVED DATA AVAILABLE
                                                                ; INTR
E85A  EE                                OUT     DX,AL
E85B  53                                PUSH    BX              ; SAVE ERROR REPORTER
E85C  83 C2 04                          ADD     DX,4            ; POINT TO LINE STATUS REGISTER
E85F  B4 01                             MOV     AH,1            ; SET RECEIVER DATA READY BIT
E861  BB 0400                           MOV     BX,0400H        ; INTR TO CHECK, INTR IDENTIFIER
E864  B9 0003                           MOV     CX,3            ; INTERRUPT ID REG 'INDEX'
E867  E8 0AF8 R                         CALL    ICT             ; PERFORM TEST FOR INTERRUPT
E86A  5B                                POP     BX              ; RESTORE ERROR INDICATOR
E86B  3C FF                             CMP     AL,0FFH         ; INTERRUPT ERROR OCCUR?
E86D  74 36                             JE      AT4             ; YES
E86F  E8 E706 R                         CALL    C5059           ; GENERATE 8259 INTERRUPT?
E872  72 33                             JC      AT5             ; NO
E874  4A                                DEC     DX
E875  4A                                DEC     DX              ; RESET INTR BY READING RECR BUFR
E876  EC                                IN      AL,DX           ; DON'T CARE ABOUT THE CONTENTS!
E877  42                                INC     DX
E878  42                                INC     DX              ; INTR ID REG
E879  E8 E719 R                         CALL    W8250C          ; WAIT FOR INTR TO CLEAR
E87C  73 03                             JNC     AT3             ; OK
E87E  E9 E948 R                         JMP     AT13            ; DIDN'T CLEAR
                                ;-----------------------------------------------------------------
                                ; TRANSMITTER HOLDING REGISTER EMPTY INTERRUPT TEST
                                ;       THIS TEST HAS BEEN MODIFIED BECAUSE THE DIFFERENT 8250'S
                                ;       THAT MAY BE USED IN PRODUCING THIS PRODUCT DO NOT FUNCTION
                                ;       THE SAME DURING THE STANDARD TEST OF THIS INTERRUPT
                                ;       (STANDARD BEING THE SAME METHOD FOR TESTING THE OTHER
                                ;       POSSIBLE 8250 INTERRUPTS).  IT IS STILL VALID FOR TESTING
                                ;       IF AN 8259 INTERRUPT IS GENERATED IN RESPONSE TO THE 8250
                                ;       INTERRUPT AND THAT THE 8250 INTERRUPT CLEARS AS IT SHOULD.
                                ;
                                ;       IF THE TRANSMITTER HOLDING REGISTER EMPTY INTERRUPT IS NOT
                                ;       GENERATED WHEN THAT INTERRUPT IS ENABLED, IT IS NOT TREATED
                                ;       AS AN ERROR.  HOWEVER, IF THE INTERRUPT IS GENERATED, IT
                                ;       MUST GENERATE AN 8259 INTERRUPT AND CLEAR PROPERLY TO PASS
                                ;       THIS TEST.
                                ;-----------------------------------------------------------------
```

```
E881  E8 E6F5 R            AT3:    CALL    SUI                 ; SET UP FOR INTERRUPTS
E884  FE C3                        INC     BL                  ; BUMP ERROR REPORTER
E886  4A                           DEC     DX                  ; POINT TO INTERRUPT ENABLE
                                                               ; REGISTER
E887  B0 02                        MOV     AL,2                ; ENABLE XMITTER HOLDING REG EMPTY
                                                               ; INTR
E889  EE                           OUT     DX,AL
E88A  EB 00                        JMP     $+2                 ; I/O DELAY
E88C  42                           INC     DX                  ; INTR IDENTIFICATION REG
E88D  2B C9                        SUB     CX,CX
E88F  EC                   AT31:   IN      AL,DX               ; READ IT
E890  3C 02                        CMP     AL,2                ; XMITTER HOLDING REG EMPTY INTR?
E892  74 04                        JE      AT32                ; YES
E894  E2 F9                        LOOP    AT31
E896  EB 11                        JMP     SHORT AT6           ; THE INTR DIDN'T OCCUR - TRY NEXT
                                                               ; TEST
E898                       AT32:                               ; THE INTR DID OCCUR
E898  E8 E706 R                    CALL    C5059               ; GENERATE 8259 INTERRUPT?
E89B  72 0A                        JC      AT5                 ; NO
E89D  E8 E719 R                    CALL    W8250C              ; WAIT FOR THE INTERRUPT TO CLEAR
                                                               ; (IT SHOULD ALREADY BE CLEAR
                                                               ; BECAUSE 'ICT' READ THE INTR ID
                                                               ; REG)
E8A0  73 07                        JNC     AT6                 ; IT CLEARED
E8A2  E9 E948 R                    JMP     AT13                ; ERROR
E8A5  EB 7E                AT4:    JMP     SHORT AT11          ; AVOID OUT OF RANGE JUMPS
E8A7  EB 7A                AT5:    JMP     SHORT AT10
                           ;----------------------------------------------------------------
                           ; RECEIVER LINE STATUS INTERRUPT TEST
                           ;       THERE ARE 4 BITS WHICH COULD GENERATE THIS INTERRUPT.
                           ;       EACH ONE IS TESTED INDIVIDUALLY.
                           ;       WHEN:   AH      TESTING
                           ;               --      -------
                           ;                2      OVERRUN
                           ;                4      PARITY
                           ;                8      FRAMING
                           ;               10H     BREAK INTR
                           ;----------------------------------------------------------------
E8A9  4A                   AT6:    DEC     DX                  ; POINT TO INTERRUPT ENABLE
                                                               ; REGISTER
E8AA  B0 04                        MOV     AL,4                ; ENABLE RECEIVER LINE STATUS INTR
E8AC  EE                           OUT     DX,AL
E8AD  83 C2 04                     ADD     DX,4                ; POINT TO LINE STATUS REGISTER
E8B0  B9 0003                      MOV     CX,3                ; INTR ID REG 'INDEX'
E8B3  BD 0004                      MOV     BP,4                ; LOOP COUNTER
E8B6  B4 02                        MOV     AH,2                ; INITIAL BIT TO BE TESTED
E8B8  E8 E6F5 R            AT7:    CALL    SUI                 ; SET UP FOR INTERRUPTS
E8BB  FE C3                        INC     BL                  ; BUMP ERROR REPORTER
E8BD  53                           PUSH    BX                  ; SAVE IT
E8BE  BB 0601                      MOV     BX,0601H            ; INTR TO CHECK, INTR IDENTIFIER
E8C1  E8 0AF8 R                    CALL    ICT                 ; PERFORM TEST FOR INTERRUPT
E8C4  5B                           POP     BX
E8C5  24 1E                        AND     AL,00011110B        ; MASK OUT BITS THAT DON'T MATTER
E8C7  3A C4                        CMP     AL,AH               ; TEST BIT ON?
E8C9  75 5A                        JNE     AT11                ; NO
E8CB  E8 E706 R                    CALL    C5059               ; GENERATE 8259 INTERRUPT?
E8CE  72 53                        JC      AT10                ; NO
E8D0  83 EA 03                     SUB     DX,3                ; INTR ID REG
E8D3  E8 E719 R                    CALL    W8250C              ; WAIT FOR THE INTR TO CLEAR
E8D6  72 70                        JC      AT13                ; IT DIDN'T
E8D8  4D                           DEC     BP                  ; ALL FOUR BITS TESTED?
E8D9  74 07                        JE      AT8                 ; YES - GO ON TO NEXT TEST
E8DB  D0 E4                        SHL     AH,1                ; GET READY FOR NEXT BIT
E8DD  83 C2 03                     ADD     DX,3                ; LINE STATUS REGISTER
E8E0  EB D6                        JMP     AT7                 ; TEST NEXT BIT
                           ;----------------------------------------------------------------
                           ; MODEM STATUS INTERRUPT TEST
                           ;       THERE ARE 4 BITS WHICH COULD GENERATE THIS INTERRUPT.
                           ;       THEY ARE TESTED INDIVIDUALLY.
                           ;       WHEN:   AH      TESTING
                           ;               --      -------
                           ;                1      DELTA CLEAR TO SEND
                           ;                2      DELTA DATA SET READY
                           ;                4      TRAILING EDGE RING INDICATOR
                           ;                8      DELTA RECEIVE LINE SIGNAL DETECT
                           ;----------------------------------------------------------------
E8E2  83 C2 04             AT8:    ADD     DX,4                ; MODEM STATUS REGISTER
E8E5  EC                           IN      AL,DX               ; CLEAR DELTA BITS THAT MAY BE ON
                                                               ; BECAUSE OF DIFFERENCES AMONG
                                                               ; 8250'S.
E8E6  EB 00                        JMP     $+2                 ; I/O DELAY
E8E8  83 EA 05                     SUB     DX,5                ; INTERRUPT ENABLE REGISTER
E8EB  B0 08                        MOV     AL,8                ; ENABLE MODEM STATUS INTERRUPT
E8ED  EE                           OUT     DX,AL
E8EE  83 C2 05                     ADD     DX,5                ; POINT TO MODEM STATUS REGISTER
E8F1  B9 0004                      MOV     CX,4                ; INTR ID REG 'INDEX'
E8F4  BD 0004                      MOV     BP,4                ; LOOP COUNTER
E8F7  B4 01                        MOV     AH,1                ; INITIAL BIT TO BE TESTED
E8F9  E8 E6F5 R            AT9:    CALL    SUI                 ; SET UP FOR INTERRUPTS
E8FC  FE C3                        INC     BL                  ; BUMP ERROR INDICATOR
E8FE  53                           PUSH    BX                  ; SAVE IT
E8FF  BB 0001                      MOV     BX,0001H            ; INTR TO CHECK, INTR IDENTIFIER
E902  E8 0AF8 R                    CALL    ICT                 ; PERFORM TEST FOR INTERRUPT
E905  5B                           POP     BX
E906  24 0F                        AND     AL,00001111B        ; MASK OUT BITS THAT DON'T MATTER
E908  3A C4                        CMP     AL,AH               ; TEST BIT ON?
E90A  75 19                        JNE     AT11                ; NO
E90C  E8 E706 R                    CALL    C5059               ; GENERATE 8259 INTERRUPT?
E90F  72 12                        JC      AT10                ; NO
E911  83 EA 04                     SUB     DX,4                ; INTR ID REG
```

```
E914  E8 E719 R                     CALL    W8250C          ; WAIT FOR INTERRUPT TO CLEAR
E917  72 2F                         JC      AT13            ; IT DIDN'T
E919  4D                            DEC,    BP
E91A  74 0B                         JE      AT12            ; ALL FOUR BITS TESTED - GO ON
E91C  D0 E4                         SHL     AH,1            ; GET READY FOR NEXT BIT
E91E  83 C2 04                      ADD     DX,4            ; MODEM STATUS REGISTER
E921  EB D6                         JMP     AT9             ; TEST NEXT BIT
                            ;------------------------------------------------------------
                            ;       POSSIBLE 8259 INTERRUPT CONTROLLER PROBLEM
                            ;------------------------------------------------------------
E923  B3 10                 AT10:   MOV     BL,10H          ; SET ERROR REPORTER
E925  EB 24                 AT11:   JMP     SHORT AT14
                            ;------------------------------------------------------------
                            ;       SET 9600 BAUD RATE AND DEFINE DATA WORD AS HAVING 8
                            ;       BITS/WORD, 2 STOP BITS, AND ODD PARITY.
                            ;------------------------------------------------------------
E927  42                    AT12:   INC     DX              ; LINE CONTROL REGISTER
E928  E8 F085 R                     CALL    S8250
                            ;------------------------------------------------------------
                            ;       SET DATA SET CONTROL WORD TO BE IN LOOP MODE
                            ;------------------------------------------------------------
E92B  83 C2 04                      ADD     DX,4
E92E  EC                            IN      AL,DX           ; CURRENT STATE
E92F  EB 00                         JMP     $+2             ; I/O DELAY
E931  0C 10                         OR      AL,00010000B    ; SET BIT 4 OF DATA SET CONTROL REG
E933  EE                            OUT     DX,AL
E934  EB 00                         JMP     $+2             ; I/O DELAY
E936  42                            INC     DX
E937  42                            INC     DX              ; MODEM STATUS REG
E938  EC                            IN      AL,DX           ; CLEAR POSSIBLE MODEM STATUS
                                                            ; INTERRUPT WHICH COULD BE CAUSED
                                                            ; BY THE OUTPUT BITS BEING LOOPED
                                                            ; TO THE INPUT BITS
E939  EB 00                         JMP     $+2             ; I/O DELAY
E93B  83 EA 06                      SUB     DX,6            ; RECEIVER BUFFER
E93E  EC                            IN      AL,DX           ; DUMMY READ TO CLEAR DATA READY
                                                            ; BIT IF IT WENT HIGH ON WRITE TO
                                                            ; MCR
                            ;------------------------------------------------------------
                            ;       PERFORM THE LOOP BACK TEST
                            ;------------------------------------------------------------
E93F  42                            INC     DX              ; INTR ENBL REG
E940  B0 00                         MOV     AL,0            ; SET FOR INTERNAL WRAP TEST
E942  CD 84                         INT     WRAP            ; DO LOOP BACK TRANSMISSION TEST
E944  B1 00                         MOV     CL,0            ; ASSUME NO ERRORS
E946  73 05                         JNC     AT15            ; WRAP TEST PASSED
E948  80 C3 10              AT13:   ADD     BL,10H          ; ERROR INDICATOR
                            ;------------------------------------------------------------
                            ;       AN ERROR WAS ENCOUNTERED SOMEWHERE DURING THE TEST
                            ;------------------------------------------------------------
E94B  B1 01                 AT14:   MOV     CL,1            ; SET FAIL INDICATOR
                            ;------------------------------------------------------------
                            ;       HOUSEKEEPING: RE-INITIALIZE THE 8250 PORTS (THE LOOP BIT
                            ;       WILL BE RESET), DISABLE THIS DEVICE INTERRUPT, SET UP
                            ;       REGISTER BH IF AN ERROR OCCURRED, AND SET OR RESET THE
                            ;       CARRY FLAG.
                            ;------------------------------------------------------------
E94D  5A                    AT15:   POP     DX              ; GET BASE ADDRESS OF 8250 ADAPTER
E94E  53                            PUSH    BX              ; SAVE ERROR CODE
E94F  E8 0AC4 R                     CALL    I8250           ; RE-INITIALIZE 8250 PORTS
E952  5B                            POP     BX
E953  2E: 8A 25                     MOV     AH,CS:[DI]      ; GET DEVICE INTERRUPT MASK
E956  20 26 0084 R                  AND     INTR_FLAG,AH    ; CLEAR DEVICE'S INTERRUPT FLAG BIT
E95A  80 F4 FF                      XOR     AH,0FFH         ; FLIP BITS
E95D  E4 21                         IN      AL,INTA01       ; GET CURRENT INTERRUPT PORT
E95F  0A C4                         OR      AL,AH           ; DISABLE THIS DEVICE INTERRUPT
E961  E6 21                         OUT     INTA01,AL
E963  9D                            POPF                    ; RE-ESTABLISH CALLER'S INTERRUPT
                                                            ; FLAG
E964  0A C9                         OR      CL,CL           ; ANY ERRORS?
E966  74 0C                         JE      AT17            ; NO
E968  B7 24                         MOV     BH,24H          ; ASSUME MODEM ERROR
E96A  80 FE 02                      CMP     DH,2            ; OR IS IT SERIAL?
E96D  75 02                         JNE     AT16            ; IT'S MODEM
E96F  B7 23                         MOV     BH,23H          ; IT'S SERIAL PRINTER
E971  F9                    AT16:   STC                     ; SET CARRY FLAG TO INDICATE ERROR
E972  EB 01                         JMP     SHORT AT18
E974  F8                    AT17:   CLC                     ; RESET CARRY FLAG - NO ERRORS
E975  58                    AT18:   POP     AX              ; RESTORE ENTRY ENABLED INTERRUPTS
E976  E6 21                         OUT     INTA01,AL       ; DEVICE INTRS RE-ESTABLISHED
E978  1F                            POP     DS              ; RESTORE REGISTER
E979  C3                            RET
E97A                        UART    ENDP
E987                                ORG     0E987H
E987  E9 1561 R                     JMP     NEAR PTR KB_INT
                            ;-------------------------------------------
                            ; NEC_OUTPUT
                            ;       THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER
                            ;       AFTER TESTING FOR CORRECT DIRECTION AND CONTROLLER READY
                            ;       THIS ROUTINE WILL TIME OUT IF THE BYTE IS NOT ACCEPTED
                            ;       WITHIN A REASONABLE AMOUNT OF TIME, SETTING THE DISKETTE
                            ;       STATUS ON COMPLETION
                            ; INPUT
                            ;       (AH)    BYTE TO BE OUTPUT
                            ; OUTPUT
                            ;       CY = 0  SUCCESS
                            ;       CY = 1  FAILURE -- DISKETTE STATUS UPDATED
                            ;               IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE
                            ;               LEVEL HIGHER THAN THE CALLER OF NEC_OUTPUT
                            ;               THIS REMOVES THE REQUIREMENT OF TESTING AFTER EVERY
                            ;               CALL OF NEC_OUTPUT
                            ;       (AL) DESTROYED
                            ;-------------------------------------------
```

```
E98A                          NEC_OUTPUT      PROC    NEAR
E98A  52                              PUSH    DX                ; SAVE REGISTERS
E98B  51                              PUSH    CX
E98C  BA 00F4                         MOV     DX,NEC_STAT       ; STATUS PORT
E98F  33 C9                           XOR     CX,CX             ; COUNT FOR TIME OUT
E991  EC                      J23:    IN      AL,DX             ; GET STATUS
E992  A8 40                           TEST    AL,DIO            ; TEST DIRECTION BIT
E994  74 0C                           JZ      J25               ; DIRECTION OK
E996  E2 F9                           LOOP    J23
E998                          J24:                              ; TIME_ERROR
E998  80 0E 0041 R 80                 OR      DISKETTE_STATUS,TIME_OUT
E99D  59                              POP     CX
E99E  5A                              POP     DX                ; SET ERROR CODE AND RESTORE REGS
E99F  58                              POP     AX                ; DISCARD THE RETURN ADDRESS
E9A0  F9                              STC                       ; INDICATE ERROR TO CALLER
E9A1  C3                              RET
E9A2  33 C9                   J25:    XOR     CX,CX             ; RESET THE COUNT
E9A4  EC                      J26:    IN      AL,DX             ; GET THE STATUS
E9A5  A8 80                           TEST    AL,RQM            ; IS IT READY?
E9A7  75 04                           JNZ     J27               ; YES, GO OUTPUT
E9A9  E2 F9                           LOOP    J26               ; COUNT DOWN AND TRY AGAIN
E9AB  EB EB                           JMP     J24               ; ERROR CONDITION
E9AD                          J27:                              ; OUTPUT
E9AD  8A C4                           MOV     AL,AH             ; GET BYTE TO OUTPUT
E9AF  42                              INC     DX                ; DATA PORT IS 1 GREATER THAN
                                                                ; STATUS PORT
E9B0  EE                              OUT     DX,AL             ; OUTPUT THE BYTE
E9B1  59                              POP     CX                ; RECOVER REGISTERS
E9B2  5A                              POP     DX
E9B3  C3                              RET                       ; CY = 0 FROM TEST INSTRUCTION
E9B4                          NEC_OUTPUT      ENDP
                              ;-----------------------------------------
                              ; GET_PARM
                              ;  THIS ROUTINE FETCHES THE INDEXED POINTER FROM
                              ;  THE DISK_BASE BLOCK POINTED AT BY THE DATA
                              ;  VARIABLE DISK_POINTER
                              ; A BYTE FROM THAT TABLE IS THEN MOVED INTO AH,
                              ;  THE INDEX OF THAT BYTE BEING THE PARM IN BX
                              ; ENTRY --
                              ;   BL = INDEX OF BYTE TO BE FETCHED * 2
                              ;        IF THE LOW BIT OF BL IS ON, THE BYTE IS IMMEDIATELY
                              ;        OUTPUT TO THE NEC CONTROLLER
                              ; EXIT --
                              ;   AH = THAT BYTE FROM BLOCK
                              ;   BX = DESTROYED
                              ;-------------------------------------------
E9B4                          GET_PARM        PROC    NEAR
E9B4  1E                              PUSH    DS                ; SAVE SEGMENT
E9B5  56                              PUSH    SI                ; SAVE REGISTER
E9B6  2B C0                           SUB     AX,AX             ; ZERO TO AX
E9B8  32 FF                           XOR     BH,BH             ; ZERO BH
E9BA  8E D8                           MOV     DS,AX
                                      ASSUME  DS:ABS0
E9BC  C5 36 0078 R                    LDS     SI,DISK_POINTER   ; POINT TO BLOCK
E9C0  D1 EB                           SHR     BX,1              ; DIVIDE BX BY 2, AND SET FLAG FOR
                                                                ; EXIT
E9C2  9C                              PUSHF                     ; SAVE OUTPUT BIT
E9C3  8A 20                           MOV     AH,[SI+BX]        ; GET THE BYTE
E9C5  83 FB 01                        CMP     BX,1              ; IS THIS THE PARM WITH DMA
                                                                ; INDICATOR
E9C8  75 05                           JNZ     J27_1
E9CA  80 CC 01                        OR      AH,1              ; TURN ON NO DMA BIT
E9CD  EB 0C                           JMP     SHORT J27_2
E9CF  83 FB 0A                J27_1:  CMP     BX,10             ; MOTOR STARTUP DELAY?
E9D2  75 07                           JNE     J27_2
E9D4  80 FC 04                        CMP     AH,4              ; GREATER THAN OR EQUAL TO 1/2 SEC?
E9D7  7D 02                           JGE     J27_2             ; YES, OKAY
E9D9  B4 04                           MOV     AH,4              ; NO, FORCE 1/2 SECOND DELAY
E9DB  9D                      J27_2:  POPF                      ; GET OUTPUT BIT
E9DC  5E                              POP     SI                ; RESTORE REGISTER
E9DD  1F                              POP     DS                ; RESTORE SEGMENT
                                      ASSUME  DS:DATA
E9DE  72 AA                           JC      NEC_OUTPUT        ; IF FLAG SET, OUTPUT TO CONTROLLER
E9E0  C3                              RET                       ; RETURN TO CALLER
E9E1                          GET_PARM        ENDP
                              ;--------------------------------------------------
                              ; BOUND_SETUP
                              ;      THIS ROUTINE SETS UP BUFFER ADDRESSING FOR READ/WRITE/VERIFY
                              ;      OPERATIONS.
                              ; INPUT
                              ;       ES HAS ORIGINAL BUFFER SEGMENT VALUE
                              ;       BP POINTS AT BASE OF SAVED PARMETERS ON STACK
                              ; OUTPUT
                              ;       ES HAS SEGMENT WHICH WILL ALLOW 64K ACCESS.  THE
                              ;       COMBINATION ES:DI  AND DS:SI POINT TO THE BUFFER. THIS
                              ;       CALCULATED ADDRESS WILL ALWAYS ACCESS 64K OF MEMORY.
                              ;       BX DESTOYED
                              ;--------------------------------------------------
```

```
E9E1                           BOUND_SETUP       PROC    NEAR
E9E1 51                                PUSH    CX              ; SAVE REGISTERS
E9E2 8B 5E 0C                          MOV     BX,[BP+12]      ; GET OFFSET OF BUFFER FROM STACK
E9E5 53                                PUSH    BX              ; SAVE OFFSET TEMPORARILY
E9E6 B1 04                             MOV     CL,4            ; SHIFT COUNT
E9E8 D3 EB                             SHR     BX,CL           ; SHIFT OFFSET FOR NEW SEGMENT
                                                               ; VALUE
E9EA 8C C1                             MOV     CX,ES           ; PUT ES IN REGISTER SUITABLE FOR
                                                               ; ADDING TO
E9EC 03 CB                             ADD     CX,BX           ; GET NEW VALUE FOR ES
E9EE 8E C1                             MOV     ES,CX           ; UPDATE THE ES REGISTER
E9F0 5B                                POP     BX              ; RECOVER ORIGINAL OFFSET
E9F1 81 E3 000F                        AND     BX,0000FH       ; NEW OFFSET
E9F5 8B F3                             MOV     SI,BX           ; DS:SI POINT AT BUFFER
E9F7 8B FB                             MOV     DI,BX           ; ES:DI POINT AT BUFFER
E9F9 59                                POP     CX
E9FA C3                                RET
E9FB                           BOUND_SETUP       ENDP
                               ;----------------------------------------------------
                               ; SEEK
                               ;       THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE
                               ;       TO THE NAMED TRACK.  IF THE DRIVE HAS NOT BEEN ACCESSED
                               ;       SINCE THE DRIVE RESET COMMAND WAS ISSUED, THE DRIVE WILL BE
                               ;       RECALIBRATED.
                               ; INPUT
                               ;       (DL) = DRIVE TO SEEK ON
                               ;       (CH) = TRACK TO SEEK TO
                               ; OUTPUT
                               ;       CY = 0 SUCCESS
                               ;       CY = 1 FAILURE -- DISKETTE_STATUS SET ACCORDINGLY
                               ;       (AX) DESTROYED
                               ;-------------------------------------------------------
E9FB                           SEEK    PROC    NEAR
E9FB 56                                PUSH    SI               ; SAVE REGISTER
E9FC 53                                PUSH    BX               ; SAVE REGISTER
E9FD 51                                PUSH    CX
E9FE BE 0074 R                         MOV     SI,OFFSET TRACK0 ; BASE OF CURRENT HEAD POSITIONS
EA01 B0 01                             MOV     AL,1             ; ESTABLISH MASK FOR RECAL
EA03 8A CA                             MOV     CL,DL            ; USE DRIVE AS A SHIFT COUNT
EA05 81 E1 00FF                        AND     CX,0FFH          ; MASK OFF HIGH BYTE
EA09 03 F1                             ADD     SI,CX            ; POINT SI AT CORRECT DRIVE
EA0B D2 C0                             ROL     AL,CL            ; GET MASK FOR DRIVE
                               ;------ SI CONTAINS OFFSET FOR CORRECT DRIVE, AL CONTAINS BIT MASK
                               ;       IN POSITION 0,1 OR 2
EA0D 59                                POP     CX               ; RESTORE PARAMETER REGISTER
EA0E BB EA66 R                         MOV     BX,OFFSET J32    ; SET UP ERROR RECOVERY ADDRESS
EA11 53                                PUSH    BX               ; NEEDED FOR ROUTINE NEC_OUTPUT
EA12 84 06 003E R                      TEST    SEEK_STATUS,AL   ; TEST DRIVE FOR RECAL
EA16 75 1B                             JNZ     J28              ; NO_RECAL
EA18 08 06 003E R                      OR      SEEK_STATUS,AL   ; TURN ON THE NO RECAL BIT IN FLAG
EA1C 80 3C 00                          CMP     BYTE PTR[SI],0   ; LAST REFERENCED TRACK=0?
EA1F 74 12                             JZ      J28              ; YES IGNORE RECAL
EA21 B4 07                             MOV     AH,07H           ; RECALIBRATE COMMAND
EA23 E8 E98A R                         CALL    NEC_OUTPUT
EA26 8A E2                             MOV     AH,DL            ; RECAL REQUIRED ON DRIVE IN DL
EA28 E8 E98A R                         CALL    NEC_OUTPUT       ; OUTPUT THE DRIVE NUMBER
                               ;------ HEAD IS MOVING TO CORRECT TRACK
EA2B E8 EA6F R                         CALL    CHK_STAT_2       ; GET THE STATUS OF RECALIBRATE
EA2E 72 39                             JC      J32_2            ; SEEK_ERROR
EA30 C6 04 00                          MOV     BYTE PTR[SI],0
                               ;----- DRIVE IS IN SYNCH WITH CONTROLLER, SEEK TO TRACK
EA33 8A 04                     J28:    MOV     AL,BYTE PTR[SI]  ; GET THE PCN
EA35 2A C5                             SUB     AL,CH            ; GET SEEK_WAIT VALUE
EA37 74 2C                             JZ      J31_1            ; ALREADY ON CORRECT TRACK
EA39 B4 0F                             MOV     AH,0FH           ; SEEK COMMAND TO NEC
EA3B E8 E98A R                         CALL    NEC_OUTPUT
EA3E 8A E2                             MOV     AH,DL            ; DRIVE NUMBER
EA40 E8 E98A R                         CALL    NEC_OUTPUT
EA43 8A E5                             MOV     AH,CH            ; TRACK NUMBER
EA45 E8 E98A R                         CALL    NEC_OUTPUT
EA48 E8 EA6F R                         CALL    CHK_STAT_2       ; GET ENDING INTERRUPT AND SENSE
                                                                ; STATUS
                               ;----- WAIT FOR HEAD SETTLE
EA4B 9C                                PUSHF                    ; SAVE STATUS FLAGS
EA4C 51                                PUSH    CX               ; SAVE REGISTER
EA4D B3 12                             MOV     BL,18            ; HEAD SETTLE PARAMETER
EA4F E8 E9B4 R                         CALL    GET_PARM
EA52                           J29:                             ; HEAD_SETTLE
EA52 B9 0226                           MOV     CX,550           ; 1 MS LOOP
EA55 0A E4                             OR      AH,AH            ; TEST FOR TIME EXPIRED
EA57 74 06                             JZ      J31
EA59 E2 FE                     J30:    LOOP    J30              ; DELAY FOR 1 MS
EA5B FE CC                             DEC     AH               ; DECREMENT THE COUNT
EA5D EB F3                             JMP     J29              ; DO IT SOME MORE
EA5F 59                        J31:    POP     CX               ; RESTORE REGISTER
EA60 9D                                POPF
EA61 72 06                             JC      J32_2
EA63 88 2C                             MOV     BYTE PTR[SI],CH
EA65 5B                        J31_1:  POP     BX               ; GET RID OF DUMMY RETURN
EA66                           J32:                             ; SEEK_ERROR
EA66 5B                                POP     BX               ; RESTORE REGISTER
EA67 5E                                POP     SI               ; UPDATE CORRECT
EA68 C3                                RET                      ; RETURN TO CALLER
EA69 C6 04 FF                  J32_2:  MOV     BYTE PTR[SI],0FFH ; UNKNOWN STATUS ABOUT SEEK
                                                                ; OPERATION
EA6C 5B                                POP     BX               ; GET RID OF DUMMY RETURN
EA6D EB F7                             JMP     SHORT J32
EA6F                           SEEK    ENDP
```

```
                               ;--------------------------------------------------
                               ; CHK_STAT_2
                               ;       THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER
                               ;       A RECALIBRATE, SEEK, OR RESET TO THE ADAPTER.
                               ;       THE INTERRUPT IS WAITED FOR, THE INTERRUPT STATUS SENSED,
                               ;       AND THE RESULT RETURNED TO THE CALLER.
                               ; INPUT
                               ;       NONE
                               ; OUTPUT
                               ;       CY = 0 SUCCESS
                               ;       CY = 1 FAILURE -- ERROR IS IN DISKETTE_STATUS
                               ;       (AX) DESTROYED
                               ;--------------------------------------------------
EA6F                           CHK_STAT_2      PROC    NEAR
EA6F  53                               PUSH    BX              ; SAVE REGISTERS
EA70  56                               PUSH    SI
EA71  33 DB                            XOR     BX,BX           ; NUMBER OF SENSE INTERRUPTS TO
                                                               ; ISSUE
EA73  BE EA88 R                        MOV     SI,OFFSET J33_3 ; SET UP DUMMY RETURN FROM
                                                               ; NEC_OUTPUT
EA76  56                               PUSH    SI              ; PUT ON STACK
EA77  B4 08                    J33_2:  MOV     AH,08H          ; SENSE INTERUPT STATUS
EA79  E8 E98A R                        CALL    NEC_OUTPUT      ; ISSUE SENSE INTERUPT STATUS
EA7C  E8 EAA0 R                        CALL    RESULTS         ;
EA7F  72 10                            JC      J35             ; NEC TIME OUT, FLAGS SET IN
                                                               ; RESULTS
EA81  A0 0042 R                        MOV     AL,NEC_STATUS   ; GET STATUS
EA84  A8 20                            TEST    AL,SEEK_END     ; IS SEEK OR RECAL OPERATION DONE?
EA86  75 0D                            JNZ     J35_1           ; JUMP IF EXECUTION OF SEEK OR
                                                               ; RECAL DONE
EA88  4B                       J33_3:  DEC     BX              ; DEC LOOP COUNTER
EA89  75 EC                            JNZ     J33_2           ; DO ANOTHER LOOP
EA8B  80 0E 0041 R 80                  OR      DISKETTE_STATUS,TIME_OUT
EA90  F9                       J34:    STC                     ; RETURN ERROR INDICATION FOR
                                                               ; CALLER
EA91  5E                       J35:    POP     SI              ; RESTORE REGISTERS
EA92  5E                               POP     SI
EA93  5B                               POP     BX
EA94  C3                               RET
                               ;-----SEEK END HAS OCCURED, CHECK FOR NORMAL TERMINATION
EA95  24 C0                    J35_1:  AND     AL,0C0H         ; MASK NORMAL TERMINATION BITS
EA97  74 F8                            JZ      J35             ; JUMP IF NORMAL TERMINATION
EA99  80 0E 0041 R 40                  OR      DISKETTE_STATUS,BAD_SEEK
EA9E  EB F0                            JMP     J34
EAA0                           CHK_STAT_2      ENDP
                               ;-----------------------------------------------
                               ; RESULTS
                               ;       THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER
                               ;       HAS TO SAY FOLLOWING AN INTERRUPT.
                               ;       IT IS ASSUMED THAT THE NEC DATA PORT = NEC STATUS PORT + 1.
                               ; INPUT
                               ;       NONE
                               ; OUTPUT
                               ;       CY = 0  SUCCESSFUL TRANSFER
                               ;       CY = 1  FAILURE -- TIME OUT IN WAITING FOR STATUS
                               ;       NEC_STATUS AREA HAS STATUS BYTE LOADED INTO IT
                               ;       (AH) DESTROYED
                               ;-----------------------------------------------
EAA0                           RESULTS PROC    NEAR
EAA0  FC                               CLD
EAA1  BF 0042 R                        MOV     DI,OFFSET NEC_STATUS ; POINTER TO DATA AREA
EAA4  51                               PUSH    CX              ; SAVE COUNTER
EAA5  52                               PUSH    DX
EAA6  53                               PUSH    BX
EAA7  B3 07                            MOV     BL,7            ; MAX STATUS BYTES
                               ;------ WAIT FOR REQUEST FOR MASTER
EAA9                           J38:                            ; INPUT_LOOP
EAA9  33 C9                            XOR     CX,CX           ; COUNTER
EAAB  BA 00F4                          MOV     DX,NEC_STAT     ; STATUS PORT
EAAE                           J39:                            ; WAIT FOR MASTER
EAAE  EC                               IN      AL,DX           ; GET STATUS
EAAF  A8 80                            TEST    AL,080H         ; MASTER READY
EAB1  75 0C                            JNZ     J40A            ; TEST_DIR
EAB3  E2 F9                            LOOP    J39             ; WAIT_MASTER
EAB5  80 0E 0041 R 80                  OR      DISKETTE_STATUS,TIME_OUT
EABA                           J40:                            ; RESULTS_ERROR
EABA  F9                               STC                     ; SET ERROR RETURN
                               ;------ RESULT OPERATION IS DONE
EABB  5B                       J44:    POP     BX
EABC  5A                               POP     DX
EABD  59                               POP     CX
EABE  C3                               RET
                               ;------ TEST THE DIRECTION BIT
EABF  EC                       J40A:   IN      AL,DX           ; GET STATUS REG AGAIN
EAC0  A8 40                            TEST    AL,040H         ; TEST DIRECTION BIT
EAC2  75 07                            JNZ     J42             ; OK TO READ STATUS
EAC4                           J41:                            ; NEC_FAIL
EAC4  80 0E 0041 R 20                  OR      DISKETTE_STATUS,BAD_NEC
EAC9  EB EF                            JMP     J40             ; RESULTS_ERROR
                               ;------ READ IN THE STATUS
EACB                           J42:                            ; INPUT_STAT
EACB  42                               INC     DX              ; POINT AT DATA PORT
EACC  EC                               IN      AL,DX           ; GET THE DATA
EACD  88 05                            MOV     [DI],AL         ; STORE THE BYTE
EACF  47                               INC     DI              ; INCREMENT THE POINTER
EAD0  B9 000A                          MOV     CX,10           ; LOOP TO KILL TIME FOR NEC
EAD3  E2 FE                    J43:    LOOP    J43
EAD5  4A                               DEC     DX              ; POINT AT STATUS PORT
EAD6  EC                               IN      AL,DX           ; GET STATUS
EAD7  A8 10                            TEST    AL,010H         ; TEST FOR NEC STILL BUSY
EAD9  74 E0                            JZ      J44             ; RESULTS DONE
EADB  FE CB                            DEC     BL              ; DECREMENT THE STATUS COUNTER
EADD  75 CA                            JNZ     J38             ; GO BACK FOR MORE
EADF  EB E3                            JMP     J41             ; CHIP HAS FAILED
```

```
                              ;-------------------------------------------------
                              ; NUM_TRANS
                              ;       THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT
                              ;       WERE ACTUALLY TRANSFERRED TO/FROM THE DISKETTE
                              ; INPUT
                              ;       (CH) = CYLINDER OF OPERATION
                              ;       (CL) = START SECTOR OF OPERATION
                              ; OUTPUT
                              ;       (AL) = NUMBER ACTUALLY TRANSFERRED
                              ;       NO OTHER REGISTERS MODIFIED
                              ;-------------------------------------------------
EAE1                          NUM_TRANS       PROC    NEAR
EAE1  A0 0045 R                       MOV     AL,NEC_STATUS+3 ; GET CYLINDER ENDED UP ON
EAE4  3A 46 0B                        CMP     AL,[BP+11]      ; SAME AS WE STARTED
EAE7  A0 0047 R                       MOV     AL,NEC_STATUS+5 ; GET ENDING SECTOR
EAEA  74 07                           JZ      J45             ; IF ON SAME CYL, THEN NO ADJUST
EAEC  B3 08                           MOV     BL,8
EAEE  E8 E9B4 R                       CALL    GET_PARM        ; GET EOT VALUE
EAF1  8A C4                           MOV     AL,AH           ;  INTO AL
EAF3  FE C0                   J45:    INC     AL              ; USE EOT+1 FOR CALCULATION
EAF5  2A 46 0A                        SUB     AL,[BP]+10      ; SUBTRACT START FROM END
EAF8  88 46 0E                        MOV     [BP+14],AL
EAFB  C3                              RET
EAFC                          NUM_TRANS       ENDP
EAFC                          RESULTS ENDP
                              ;-------------------------------------------------
                              ; DISABLE
                              ;       THIS ROUTINE WILL DISABLE ALL INTERRUPTS EXCEPT FOR
                              ;       INTERRUPT 6 SO WATCH DOG TIME OUT CAN OCCUR IN ERROR
                              ;       CONDITIONS.
                              ; INPUT
                              ;       NONE
                              ; OUTPUT
                              ;       NONE
                              ;       ALL REGISTERS REMAIN INTACT
                              ;-------------------------------------------------
EAFC                          DISABLE         PROC    NEAR
EAFC  50                              PUSH    AX
                              ;------ DISABLE ALL INTERRUPTS AT THE 8259 LEVEL EXCEPT DISKETTE
EAFD  E4 21                           IN      AL,INTA01       ; READ CURRENT MASK
EAFF  89 46 10                        MOV     [BP+16],AX      ; SAVE MASK ON THE SPACE ALLOCATED
                                                              ; ON THE STACK
EB02  B0 BF                           MOV     AL,0BFH         ; MASK OFF ALL INTERRUPTS EXCEPT
                                                              ; DISKETTE
EB04  E6 21                           OUT     INTA01,AL       ; OUTPUT MASK TO THE 8259
EB06  E8 E9E1 R                       CALL    BOUND_SETUP     ; SETUP REGISTERS TO ACCESS BUFFER
EB09  58                              POP     AX
EB0A  C3                              RET
EB0B                          DISABLE         ENDP
                              ;-------------------------------------------------
                              ; ENABLE
                              ;       THIS PROC ENABLES ALL INTERRUPTS.  IT ALSO SETS THE 8253 TO
                              ;       THE MODE REQUIRED FOR KEYBOARD DATA DESERIALIZATION.
                              ;       BEFORE THE LATCH FOR KEYBOARD DATA IS RESET, BIT 0 OF THE
                              ;       8255 IS READ TO DETERMINE WHETHER ANY KEYSTROKES OCCURED
                              ;       WHILE THE SYSTEM WAS MASKED OFF.
                              ; INPUT
                              ;       NONE
                              ; OUTPUT
                              ;       AL=1 MEANS A KEY WAS STRUCK DURING DISKETTE I/O.  (OR NOISE
                              ;            ON THE LINE)
                              ;       AL=0 MEANS THAT NO KEY WAS PRESSED.
                              ;       AX IS DESTROYED.  ALL OTHER REGISTERS REMAIN INTACT.
                              ;-------------------------------------------------
EB0B                          ENABLE          PROC    NEAR
EB0B  52                              PUSH    DX              ; SAVE DX
                              ;------ RETURN TIMER1 TO STATE NEEDED FOR KEYBOARD I/O
EB0C  B0 76                           MOV     AL,01110110B    ;
EB0E  E6 43                           OUT     TIM_CTL,AL
EB10  50                              PUSH    AX
EB11  58                              POP     AX              ; WAIT FOR 8253 TO INITIALIZE
                                                              ; ITSELF
EB12  B0 FF                           MOV     AL,0FFH         ; INITIAL VALUE FOR 8253
EB14  E6 41                           OUT     TIMER+1,AL      ; LSB
EB16  50                              PUSH    AX
EB17  58                              POP     AX              ; WAIT
EB18  E6 41                           OUT     TIMER+1,AL      ; MSB
                              ;------ CHECK IF ANY KEYSTROKES OCCURED DURING DISKETTE TRANSFER
EB1A  8E 46 10                        MOV     ES,[BP+16]      ; GET ORIGINAL ES VALUE FROM THE
                                                              ; STACK
EB1D  E4 62                           IN      AL,62H          ; READ PORT C OF 8255
EB1F  24 01                           AND     AL,01H          ; BIT=1 MEANS KESTROKE HAS OCCURED
EB21  50                              PUSH    AX              ; SAVE IT ON THE STACK
                              ;------ ENABLE NMI INTERRUPTS
EB22  E4 A0                           IN      AL,NMI_PORT     ; RESET LATCH
EB24  B0 80                           MOV     AL,80H          ; MASK TO ENABLE NMI
EB26  E6 A0                           OUT     NMI_PORT,AL     ; ENABLE NMI
                              ;-----  ENABLE ALL INTERRUPTS WHICH WERE ENABLED BEFORE TRANSFER
EB28  8B 46 10                        MOV     AX,[BP+16]      ; GET MASK FROM THE STACK
EB2B  E6 21                           OUT     INTA01,AL
EB2D  58                              POP     AX              ; PASS BACK KEY STROKE FLAG
EB2E  5A                              POP     DX
EB2F  FB                              STI
EB30  C3                              RET
EB31                          ENABLE          ENDP
```

```
                                  ;------------------------------------------------
                                  ;CLOCK_WAIT
                                  ;       THIS PROCEDURE IS CALLED WHEN THE TIME OF DAY
                                  ;       IS BEING UPDATED.  IT WAITS IF TIMERO IS ALMOST
                                  ;       READY TO WRAP UNTIL IT IS SAFE TO READ AN ACCURATE
                                  ;       TIMER1.
                                  ;INPUT
                                  ;       NONE.
                                  ;OUTPUT
                                  ;       NONE. AX IS DESTROYED.
                                  ;------------------------------------------------
EB31                              CLOCK_WAIT      PROC    NEAR
EB31  32 C0                               XOR     AL,AL           ; READ MODE TIMERO FOR 8253
EB33  E6 43                               OUT     TIM_CTL,AL      ; OUTPUT TO THE 8253
EB35  50                                  PUSH    AX
EB36  58                                  POP     AX              ; WAIT FOR 8253 TO INITIALIZE
                                                                  ; ITSELF
EB37  E4 40                               IN      AL,TIMERO       ; READ LEAST SIGNIFICANT BYTE
EB39  86 C4                               XCHG    AL,AH           ; SAVE IT
EB3B  E4 40                               IN      AL,TIMERO       ; READ MOST SIGNIFICANT BYTE
EB3D  86 C4                               XCHG    AL,AH           ; REARRANGE FOR PROPER ORDER
EB3F  3D 012C                             CMP     AX,THRESHOLD    ; IS TIMERO CLOSE TO WRAPPING?
EB42  72 ED                               JC      CLOCK_WAIT      ; JUMP IF CLOCK IS WITHIN THRESHOLD
EB44  C3                                  RET                     ; OK TO READ TIMER1
EB45                              CLOCK_WAIT      ENDP
                                  ;------------------------------------------------------------
                                  ;GET_DRIVE
                                  ;       THIS ROUTINE WILL CALCULATE A BIT MASK FOR THE DRIVE WHICH
                                  ;       IS SELECTED BY THE CURRENT INT 13 CALL.  THE DRIVE SELECTED
                                  ;       CORRESPONDS TO THE BIT IN THE MASK,  I.E. DRIVE ZERO
                                  ;       CORRESPONDS TO BIT ZERO AND A 01H IS RETURNED. THE BIT IS
                                  ;       CALCULATED BY ACCESSING THE PARAMETERS PASSED TO INT 13
                                  ;       WHICH WERE SAVED ON THE STACK.
                                  ;INPUT
                                  ;       BYTE PTR[BP] MUST POINT TO DRIVE FOR SELECTION.
                                  ;OUPTUT
                                  ;       AL CONTAINS THE BIT MASK.  ALL OTHER REGISTERS ARE INTACT
                                  ;------------------------------------------------------------
EB45                              GET_DRIVE       PROC    NEAR
EB45  51                                  PUSH    CX              ; SAVE REGISTER.
EB46  8A 4E 00                            MOV     CL,BYTE PTR[BP] ; GET DRIVE NUMBER
EB49  B0 01                               MOV     AL,1            ; INITIALIZE AL WITH VALUE FOR
                                                                  ; SHIFTING
EB4B  D2 E0                               SHL     AL,CL           ; SHIFT BIT POSITION BY DRIVE
                                                                  ; NUMBER (DRIVE IN RANGE 0-2)
EB4D  24 07                               AND     AL,07H          ; ONLY THREE DRIVES ARE SUPPORTED.
                                                                  ;  RANGE CHECK
EB4F  59                                  POP     CX              ; RESTORE REGISTERS
EB50  C3                                  RET
EB51                              GET_DRIVE       ENDP
                                  ;------------------------------------------------
                                  ; THIS ROUTINE  CHECKS OPTIONAL ROM MODULES (CHECKSUM
                                  ; FOR MODULES FROM C0000->D0000, CRC CHECK FOR CARTRIDGES
                                  ; (D0000->F0000)
                                  ; IF CHECK IS OK, CALLS INIT/TEST CODE IN MODULE
                                  ;    MFG ERROR CODE= 25XX (XX=MSB OF SEGMENT IN ERROR)
                                  ;------------------------------------------------
EB51                              ROM_CHECK       PROC    NEAR
EB51  2B F6                               SUB     SI,SI           ; SET SI TO POINT TO BEGINNING
                                                                  ; (REL. TO DS)
EB53  2A C0                               SUB     AL,AL           ; ZERO OUT AL
EB55  8A 67 02                            MOV     AH,[BX+2]       ; GET LENGTH INDICATOR
EB58  D1 E0                               SHL     AX,1            ; FORM COUNT
EB5A  50                                  PUSH    AX              ; SAVE COUNT
EB5B  81 FA D000                          CMP     DX,0D000H       ; SEE IF POINTER IS BELOW D000
EB5F  9C                                  PUSHF                   ; SAVE RESULTS
EB60  B1 04                               MOV     CL,4            ; ADJUST
EB62  D3 E8                               SHR     AX,CL           ;
EB64  03 D0                               ADD     DX,AX           ; SET POINTER TO NEXT MODULE
EB66  9D                                  POPF                    ; RECOVER FLAGS FROM POINTER RANGE
                                                                  ; CHECK
EB67  59                                  POP     CX              ; RECOVER COUNT IN CX REGISTER
EB68  52                                  PUSH    DX              ; SAVE POINTER
EB69  7C 07                               JL      ROM_1           ; DO ARITHMETIC CHECKSUM IF BELOW
                                                                  ; D0000
EB6B  E8 FE71 R                           CALL    CRC_CHECK       ; DO CRC CHECK
EB6E  74 2B                               JZ      ROM_CHECK_1     ; PROCEED IF OK
EB70  EB 05                               JMP     SHORT ROM_2     ; ELSE POST ERROR
EB72  E8 FEEB R                   ROM_1:  CALL    ROS_CHECKSUM    ; DO ARITHMETIC CHECKSUM
EB75  74 24                               JZ      ROM_CHECK_1     ; PROCEED IF OK
EB77  BA 1626                     ROM_2:  MOV     DX,1626H        ; POSITION CURSOR, ROW 22, COL 38
EB7A  B4 02                               MOV     AH,2
EB7C  B7 07                               MOV     BH,7
EB7E  CD 10                               INT     10H
EB80  8C DA                               MOV     DX,DS           ; RECOVER DATA SEG
EB82  8A C6                               MOV     AL,DH           ;
EB84  E8 18A9 R                           CALL    XPC_BYTE        ; DISPLAY MSB OF DATA SEG
EB87  8A DE                               MOV     BL,DH           ; FORM XX VALUE OF ERROR CODE
EB89  B7 25                               MOV     BH,25H          ; FORM 25 PORTION
EB8B  80 FE D0                            CMP     DH,0D0H         ; IN CARTRIDGE SPACE?
EB8E  BE 003B R                           MOV     SI,OFFSET CART_ERR
EB91  7D 03                               JGE     ROM_CHECK_0     ;
EB93  BE 003A R                           MOV     SI,OFFSET ROM_ERR
EB96                              ROM_CHECK_0:
EB96  E8 09BC R                           CALL    E_MSG           ; GO ERROR ROUTINE
EB99  EB 16                               JMP     SHORT ROM_CHECK_END   ; AND EXIT
EB9B                              ROM_CHECK_1:
EB9B  B8 ---- R                           MOV     AX,XXDATA       ; SET ES TO POINT TO XXDATA AREA
EB9E  8E C0                               MOV     ES,AX           ;
EBA0  26: C7 06 0014 R 0003               MOV     ES:IO_ROM_INIT,0003H  ; LOAD OFFSET
EBA7  26: 8C 1E 0016 R                    MOV     ES:IO_ROM_SEG,DS      ; LOAD SEGMENT
EBAC  26: FF 1E 0014 R                    CALL    DWORD PTR ES:IO_ROM_INIT ; CALL INIT./TEST ROUTINE
```

```
EBB1                         ROM_CHECK_END:
EBB1  5A                            POP     DX              ; RECOVER POINTER
EBB2  C3                            RET                     ; RETURN TO CALLER
EBB3                         ROM_CHECK        ENDP
                             ;-- INT 13 ----------------------------------------------------
                             ; DISKETTE I/O
                             ;      THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4" DISKETTE DRIVES
                             ; INPUT
                             ;      (AH)=0  RESET DISKETTE SYSTEM
                             ;              HARD RESET TO NEC, PREPARE COMMAND, RECAL REQD ON
                             ;              ALL DRIVES
                             ;      (AH)=1  READ THE STATUS OF THE SYSTEM INTO (AL)
                             ;              DISKETTE_STATUS FROM LAST OP'N IS USED
                             ;      REGISTERS FOR READ/WRITE/VERIFY/FORMAT
                             ;      (DL) - DRIVE NUMBER (0-3 ALLOWED, VALUE CHECKED)
                             ;      (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
                             ;      (CH) - TRACK NUMBER (0-39, NOT VALUE CHECKED)
                             ;      (CL) - SECTOR NUMBER (1-8, NOT VALUE CHECKED, NOT USED FOR
                             ;             FORMAT)
                             ;      (AL) - NUMBER OF SECTORS ( MAX = 8, NOT VALUE CHECKED, NOT
                             ;             USED FOR FORMAT, HOWEVER, CANNOT BE ZERO!!!)
                             ;      (ES:BX) - ADDRESS OF BUFFER ( NOT REQUIRED FOR VERIFY)
                             ;
                             ;      (AH)=2  READ THE DESIRED SECTORS INTO MEMORY
                             ;      (AH)=3  WRITE THE DESIRED SECTORS FROM MEMORY
                             ;      (AH)=4  VERIFY THE DESIRED SECTORS
                             ;      (AH)=5  FORMAT THE DESIRED TRACK
                             ;              FOR THE FORMAT OPERATION, THE BUFFER POINTER
                             ;              (ES,BX) MUST POINT TO THE COLLECTION OF DESIRED
                             ;              ADDRESS FIELDS FOR THE TRACK.  EACH FIELD IS
                             ;              COMPOSED OF 4 BYTES, (C,H,R,N), WHERE
                             ;              C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER,
                             ;              N= NUMBER OF BYTES PER SECTOR (00=128, 01=256,
                             ;              02=512, 03=1024,). THERE MUST BE ONE ENTRY FOR
                             ;              EVERY SECTOR ON THE TRACK. THIS INFORMATION IS USED
                             ;              TO FIND THE REQUESTED SECTOR DURING READ/WRITE
                             ;              ACCESS.
                             ; DATA VARIABLE -- DISK_POINTER
                             ;     DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
                             ; OUTPUT
                             ;      AH = STATUS OF OPERATION
                             ;              STATUS BITS ARE DEFINED IN THE EQUATES FOR
                             ;              DISKETTE_STATUS VARIABLE IN THE DATA SEGMENT OF
                             ;              THIS MODULE
                             ;      CY = 0  SUCCESSFUL OPERATION (AH=0 ON RETURN)
                             ;      CY = 1  FAILED OPERATION (AH HAS ERROR REASON)
                             ;      FOR READ/WRITE/VERIFY
                             ;              DS,BX,DX,CH,CL PRESERVED
                             ;              AL = NUMBER OF SECTORS ACTUALLY READ
                             ;              **** AL MAY NOT BE CORRECT IF TIME OUT ERROR OCCURS
                             ;      NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE
                             ;              APPROPRIATE ACTION IS TO RESET THE DISKETTE, THEN
                             ;              RETRY THE OPERATION. ON READ ACCESSES, NO MOTOR
                             ;              START DELAY IS TAKEN, SO THAT THREE RETRIES ARE
                             ;              REQUIRED ON READS TO ENSURE THAT THE PROBLEM IS NOT
                             ;              DUE TO MOTOR START-UP.
                             ;-------------------------------------------------------------
                                    ASSUME  CS:CODE,DS:DATA,ES:DATA
EC59                                ORG     0EC59H
EC59                         DISKETTE_IO     PROC    FAR
EC59  FB                            STI                     ; INTERRUPTS BACK ON
EC5A  06                            PUSH    ES              ; SAVE ES
EC5B  50                            PUSH    AX              ; ALLOCATE ONE WORD OF STORAGE FOR
                                                            ; TIMER1 INITIAL VALUE
EC5C  50                            PUSH    AX              ; ALLOCATE ONE WORD ON STACK FOR
                                                            ; USE IN PROCS ENABLE AND DISABLE.
                                                            ;  WILL HOLD 8259 MASK.
EC5D  50                            PUSH    AX              ; SAVE COMMAND AND N_SECTORS
EC5E  53                            PUSH    BX              ; SAVE ADDRESS
EC5F  51                            PUSH    CX
EC60  1E                            PUSH    DS              ; SAVE SEGMENT REGISTER VALUE
EC61  56                            PUSH    SI              ; SAVE ALL REGISTERS DURING
                                                            ; OPERATION
EC62  57                            PUSH    DI
EC63  55                            PUSH    BP
EC64  52                            PUSH    DX
EC65  8B EC                         MOV     BP,SP           ; SET UP POINTER TO HEAD PARM
EC67  E8 138B R                     CALL    DDS             ; SET DS=DATA
EC6A  E8 EC90 R                     CALL    J1              ; CALL THE REST TO ENSURE DS
                                                            ; RESTORED
EC6D  B3 04                         MOV     BL,4            ; GET THE MOTOR WAIT PARAMETER
EC6F  E8 E9B4 R                     CALL    GET_PARM
EC72  88 26 0040 R                  MOV     MOTOR_COUNT,AH  ; SET THE TIMER COUNT FOR THE MOTOR
EC76  8A 26 0041 R                  MOV     AH,DISKETTE_STATUS      ; GET STATUS OF OPERATION
EC7A  88 66 0F                      MOV     [BP+15],AH      ; RETURN STATUS IN AL
EC7D  5A                            POP     DX              ; RESTORE ALL REGISTERS
EC7E  5D                            POP     BP
EC7F  5F                            POP     DI
EC80  5E                            POP     SI
EC81  1F                            POP     DS
EC82  59                            POP     CX
EC83  5B                            POP     BX              ; RECOVER OFFSET
EC84  58                            POP     AX
EC85  83 C4 04                      ADD     SP,4            ; DISCARD DUMMY SPACE FOR 8259 MASK
EC88  07                            POP     ES              ; RECOVER SEGMENT
EC89  80 FC 01                      CMP     AH,1            ; SET THE CARRY FLAG TO INDICATE
EC8C  F5                            CMC                     ;  SUCCESS OR FAILURE
EC8D  CA 0002                       RET     2               ; THROW AWAY SAVED FLAGS
```

```
EC90                           DISKETTE_IO     ENDP
EC90                           J1      PROC    NEAR
EC90  8A F0                            MOV     DH,AL            ; SAVE # SECTORS IN DH
EC92  80 26 003F R 7F                  AND     MOTOR_STATUS,07FH        ; INDICATE A READ OPERATION
EC97  0A E4                            OR      AH,AH            ; AH=0
EC99  74 27                            JZ      DISK_RESET
EC9B  FE CC                            DEC     AH               ; AH=1
EC9D  74 74                            JZ      DISK_STATUS
EC9F  C6 06 0041 R 00                  MOV     DISKETTE_STATUS,0 ; RESET THE STATUS INDICATOR
ECA4  80 FA 02                         CMP     DL,2             ; TEST FOR DRIVE IN 0-2 RANGE
ECA7  77 13                            JA      J3               ; ERROR IF ABOVE
ECA9  FE CC                            DEC     AH               ; AH=2
ECAB  74 6D                            JZ      DISK_READ
ECAD  FE CC                            DEC     AH               ; AH=3
ECAF  75 03                            JNZ     J2               ; TEST_DISK_VERF
ECB1  E9 ED3D R                        JMP     DISK_WRITE
ECB4                           J2:                              ; TEST_DISK_VERF
ECB4  FE CC                            DEC     AH               ; AH=4
ECB6  74 62                            JZ      DISK_VERF
ECB8  FE CC                            DEC     AH               ; AH=5
ECBA  74 62                            JZ      DISK_FORMAT
ECBC                           J3:                              ; BAD_COMMAND
ECBC  C6 06 0041 R 01                  MOV     DISKETTE_STATUS,BAD_CMD ; ERROR CODE, NO SECTORS
                                                                ; TRANSFERRED
ECC1  C3                               RET                      ; UNDEFINED OPERATION
ECC2                           J1      ENDP
                               ;------ RESET THE DISKETTE SYSTEM
ECC2                           DISK_RESET      PROC    NEAR
ECC2  BA 00F2                          MOV     DX,NEC_CTL       ; ADAPTER CONTROL PORT
ECC5  FA                               CLI                      ; NO INTERRUPTS
ECC6  A0 003F R                        MOV     AL,MOTOR_STATUS  ; FIND OUT IF MOTOR IS RUNNING
ECC9  24 07                            AND     AL,07H           ; DRIVE BITS
ECCB  EE                               OUT     DX,AL            ; RESET THE ADAPTER
ECCC  C6 06 003E R 00                  MOV     SEEK_STATUS,0    ; SET RECAL REQUIRED ON ALL DRIVES
ECD1  C6 06 0041 R 00                  MOV     DISKETTE_STATUS,0 ; SET OK STATUS FOR DISKETTE
ECD6  0C 80                            OR      AL,FDC_RESET     ; TURN OFF RESET
ECD8  EE                               OUT     DX,AL            ; TURN OFF THE RESET
ECD9  FB                               STI                      ; REENABLE THE INTERRUPTS
ECDA  BE ECFA R                        MOV     SI,OFFSET J4_2   ; DUMMY RETURN FOR
ECDD  56                               PUSH    SI               ; PUSH RETURN IF ERROR
                                                                ; IN NEC_OUTPUT
ECDE  B9 0010                          MOV     CX,10H           ; NUMBER OF SENSE INTERRUPTS TO
                                                                ; ISSUE
ECE1  B4 08                    J4_0:   MOV     AH,08H           ; COMMAND FOR SENSE INTERRUPT
                                                                ; STATUS
ECE3  E8 E98A R                        CALL    NEC_OUTPUT       ; OUTPUT  THE SENSE INTERRUPT
                                                                ; STATUS
ECE6  E8 EAA0 R                        CALL    RESULTS          ; GET STATUS FOLLOWING COMPLETION
                                                                ; OF RESET
ECE9  A0 0042 R                        MOV     AL,NEC_STATUS    ; IGNORE ERROR RETURN AND DO OWN
                                                                ; TEST
ECEC  3C C0                            CMP     AL,0C0H          ; TEST FOR DRIVE READY TRANSITION
ECEE  74 12                            JZ      J7               ; EVERYTHING OK
ECF0  E2 EF                            LOOP    J4_0             ; RETRY THE COMMAND
ECF2  80 0E 0041 R 20          J4_1:   OR      DISKETTE_STATUS,BAD_NEC ; SET ERROR CODE
ECF7  5E                               POP     SI
ECF8  EB 18                            JMP     SHORT J8
ECFA  BE ECFA R                J4_2:   MOV     SI,OFFSET J4_2   ; NEC_OUTPUT FAILED, RETRY THE
                                                                ; SENSE INTERRUPT
ECFD  56                               PUSH    SI               ; OFFSET OF BAD RETURN IN
                                                                ; NEC_OUTPUT
ECFE  E2 E1                            LOOP    J4_0             ; RETRY
ED00  EB F0                            JMP     SHORT J4_1
                               ;------ SEND SPECIFY COMMAND TO NEC
ED02  5E                       J7:     POP     SI               ; GET RID OF DUMMY ARGUMENT
ED03  B4 03                            MOV     AH,03H           ; SPECIFY COMMAND
ED05  E8 E98A R                        CALL    NEC_OUTPUT       ; OUTPUT THE COMMAND
ED08  B3 01                            MOV     BL,1             ; STEP RATE TIME AND HEAD UNLOAD
ED0A  E8 E9B4 R                        CALL    GET_PARM         ; OUTPUT TO THE NEC CONTROLLER
ED0D  B3 03                            MOV     BL,3             ; PARM1 HEAD LOAD AND NO DMA
ED0F  E8 E9B4 R                        CALL    GET_PARM         ;  TO THE NEC CONTROLLER
ED12                           J8:                              ; RESET_RET
ED12  C3                               RET                      ; RETURN TO CALLER
ED13                           DISK_RESET      ENDP
                               ;------ DISKETTE STATUS ROUTINE
ED13                           DISK_STATUS     PROC    NEAR
ED13  A0 0041 R                        MOV     AL,DISKETTE_STATUS
ED16  88 46 0E                         MOV     BYTE PTR[BP+14],AL ; PUT STATUS ON STACK, IT WILL
                                                                ; POP IN AL
ED19  C3                               RET
ED1A                           DISK_STATUS     ENDP
                               ;------ DISKETTE VERIFY
ED1A                           DISK_VERF       LABEL   NEAR
                               ;------ DISKETTE READ
ED1A                           DISK_READ       PROC    NEAR
ED1A                           J9:                              ; DISK_READ_CONT
ED1A  B4 46                            MOV     AH,046H          ; SET UP READ COMMAND FOR NEC
                                                                ; CONTROLLER
ED1C  EB 26                            JMP     SHORT RW_OPN     ; GO DO THE OPERATION
ED1E                           DISK_READ       ENDP
                               ;------ DISKETTE FORMAT
ED1E                           DISK_FORMAT     PROC    NEAR
ED1E  80 0E 003F R 80                  OR      MOTOR_STATUS,80H ; INDICATE A WRITE OPERATION
ED23  B4 4D                            MOV     AH,04DH          ; ESTABLISH THE FORMAT COMMAND
ED25  EB 1D                            JMP     SHORT RW_OPN     ; DO THE OPERATION
```

```
ED27                           J10:                           ; CONTINUATION OF RW_OPN FOR FMT
ED27  B3 07                            MOV     BL,7            ; GET THE
ED29  E8 E9B4 R                        CALL    GET_PARM        ;  BYTES/SECTOR VALUE TO NEC
ED2C  B3 09                            MOV     BL,9            ; GET THE
ED2E  E8 E9B4 R                        CALL    GET_PARM        ;  SECTORS/TRACK VALUE TO NEC
ED31  B3 0F                            MOV     BL,15           ; GET THE
ED33  E8 E9B4 R                        CALL    GET_PARM        ;  GAP LENGTH VALUE TO NEC
ED36  BB 0011                          MOV     BX,17           ; GET THE FILLER BYTE
ED39  53                               PUSH    BX              ; SAVE PARAMETER INDEX ON STACK
ED3A  E9 EDCD R                        JMP     J16             ;  TO THE CONTROLLER
ED3D                           DISK_FORMAT     ENDP
                               ;------ DISKETTE WRITE ROUTINE
ED3D                           DISK_WRITE      PROC    NEAR
ED3D  80 0E 003F R 80                  OR      MOTOR_STATUS,80H ; INDICATE A WRITE OPERATION
ED42  B4 45                            MOV     AH,045H         ; NEC COMMAND TO WRITE TO DISKETTE
ED44                           DISK_WRITE      ENDP
                               ;----- ALLOW WRITE ROUTINE TO FALL INTO RW_OPN
                               ;--------------------------------------
                               ; RW_OPN
                               ;       THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION
                               ;--------------------------------------
ED44                           RW_OPN  PROC    NEAR
ED44  50                               PUSH    AX              ; SAVE THE COMMAND
                               ;------ TURN ON THE MOTOR AND SELECT THE DRIVE
ED45  51                               PUSH    CX              ; SAVE THE T/S PARMS
ED46  FA                               CLI                     ; NO INTERRUPTS WHILE DETERMINING
                                                               ; MOTOR STATUS
ED47  C6 06 0040 R FF                  MOV     MOTOR_COUNT,0FFH ; SET LARGE COUNT DURING OPERATION
ED4C  E8 EB45 R                        CALL    GET_DRIVE       ; GET THE DRIVE PARAMETER FROM THE
                                                               ; STACK
ED4F  84 06 003F R                     TEST    MOTOR_STATUS,AL ; TEST MOTOR FOR OPERATING
ED53  75 1F                            JNZ     J14             ; IF RUNNING, SKIP THE WAIT
ED55  80 26 003F R F0                  AND     MOTOR_STATUS,0F0H ; TURN OFF RUNNING DRIVE
ED5A  08 06 003F R                     OR      MOTOR_STATUS,AL         ; TURN ON THE CURRENT MOTOR
ED5E  FB                               STI                     ; INTERRUPTS BACK ON
ED5F  0C 80                            OR      AL,FDC_RESET    ; NO RESET.  TURN ON MOTOR
ED61  E6 F2                            OUT     NEC_CTL,AL
                               ;------ WAIT FOR MOTOR BOTH READ AND WRITE
ED63  B3 14                            MOV     BL,20           ; GET MOTOR START TIME
ED65  E8 E9B4 R                        CALL    GET_PARM
ED68  0A E4                            OR      AH,AH           ; TEST FOR NO WAIT
ED6A                           J12:                           ; TEST_WAIT_TIME
ED6A  74 08                            JZ      J14             ; EXIT WITH TIME EXPIRED
ED6C  2B C9                            SUB     CX,CX           ; SET UP 1/8 SECOND LOOP TIME
ED6E  E2 FE                    J13:    LOOP    J13             ; WAIT FOR THE REQUIRED TIME
ED70  FE CC                            DEC     AH              ; DECREMENT TIME VALUE
ED72  EB F6                            JMP     J12             ; ARE WE DONE YET
ED74                           J14:                           ; MOTOR_RUNNING
ED74  FB                               STI                     ; INTERRUPTS BACK ON FOR BYPASS
                                                               ; WAIT
ED75  59                               POP     CX
                               ;------ DO THE SEEK OPERATION
ED76  E8 E9FB R                        CALL    SEEK            ; MOVE TO CORRECT TRACK
ED79  58                               POP     AX              ; RECOVER COMMAND
ED7A  8A FC                            MOV     BH,AH           ; SAVE COMMAND IN BH
ED7C  B6 00                            MOV     DH,0            ; SET NO SECTORS READ IN CASE OF
                                                               ; ERROR
ED7E  73 03                            JNC     J14_1           ; IF NO ERROR CONTINUE, JUMP AROUND
                                                               ; JMP
ED80  E9 EED7 R                        JMP     J17             ; CARRY SET JUMP TO MOTOR WAIT
ED83  BE EED7 R                J14_1:  MOV     SI,OFFSET J17   ; DUMMY RETURN ON STACK FOR
                                                               ; NEC_OUTPUT
ED86  56                               PUSH    SI              ;  SO THAT IT WILL RETURN TO MOTOR
                                                               ; OFF LOCATION
                               ;------ SEND OUT THE PARAMETERS TO THE CONTROLLER
ED87  E8 E98A R                        CALL    NEC_OUTPUT      ; OUTPUT THE OPERATION COMMAND
ED8A  8A 66 01                         MOV     AH,[BP+1]       ; GET THE CURRENT HEAD NUMBER
ED8D  D0 E4                            SAL     AH,1            ; MOVE IT TO BIT 2
ED8F  D0 E4                            SAL     AH,1
ED91  80 E4 04                         AND     AH,4            ; ISOLATE THAT BIT
ED94  0A E2                            OR      AH,DL           ; OR IN THE DRIVE NUMBER
ED96  E8 E98A R                        CALL    NEC_OUTPUT
                               ;------ TEST FOR FORMAT COMMAND
ED99  80 FF 4D                         CMP     BH,04DH         ; IS THIS A FORMAT OPERATION?
ED9C  75 02                            JNE     J15             ; NO. CONTINUE WITH R/W/V
ED9E  EB 87                            JMP     J10             ; IF SO, HANDLE SPECIAL
EDA0  8A E5                    J15:    MOV     AH,CH           ; CYLINDER NUMBER
EDA2  E8 E98A R                        CALL    NEC_OUTPUT
EDA5  8A 66 01                         MOV     AH,[BP+1]       ; HEAD NUMBER FROM STACK
EDA8  E8 E98A R                        CALL    NEC_OUTPUT
EDAB  8A E1                            MOV     AH,CL           ; SECTOR NUMBER
EDAD  E8 E98A R                        CALL    NEC_OUTPUT
EDB0  B3 07                            MOV     BL,7            ; BYTES/SECTOR PARM FROM BLOCK
EDB2  E8 E9B4 R                        CALL    GET_PARM        ;  TO THE NEC
EDB5  B3 08                            MOV     BL,8            ; EOT PARM FROM BLOCK
EDB7  E8 E9B4 R                        CALL    GET_PARM        ;  RETURNED IN AH
EDBA  02 4E 0E                         ADD     CL,[BP+14]      ; ADD CURRENT SECTOR TO NUMBER IN
                                                               ; TRANSFER
EDBD  FE C9                            DEC     CL              ; CURRENT_SECTOR + N_SECTORS - 1
EDBF  8A E1                            MOV     AH,CL           ; EOT PARAMETER IS THE CALCULATED
                                                               ; ONE
EDC1  E8 E98A R                        CALL    NEC_OUTPUT
EDC4  B3 0B                            MOV     BL,11           ; GAP LENGTH PARM FROM BLOCK
EDC6  E8 E9B4 R                        CALL    GET_PARM        ;  TO THE NEC
EDC9  BB 000D                          MOV     BX,13           ; DTL PARM FROM BLOCK
EDCC  53                               PUSH    BX              ; SAVE INDEX TO DISK PARAMETER ON
                                                               ; STACK
```

```
EDCD  FC                  J16:    CLD                       ; FORWARD DIRECTION
                          ;------ START TIMER1 WITH INITIAL VALUE OF FFFF
EDCE  B0 70                       MOV     AL,01110000B      ; SELECT TIMER1,LSB-MSB, MODE 0,
                                                            ; BINARY COUNTER
EDD0  E6 43                       OUT     TIM_CTL,AL        ; INITIALIZE THE COUNTER
EDD2  50                          PUSH    AX
EDD3  58                          POP     AX                ; ALLOW ENOUGH TIME FOR THE 8253 TO
                                                            ; INITIALIZE ITSELF
EDD4  B0 FF                       MOV     AL,0FFH           ; INITIAL COUNT VALUE FOR THE 8253
EDD6  E6 41                       OUT     TIMER+1,AL        ; OUTPUT LEAST SIGNIFICANT BYTE
EDD8  50                          PUSH    AX
EDD9  58                          POP     AX                ; WAIT
EDDA  E6 41                       OUT     TIMER+1,AL        ; OUTPUT MOST SIGNIFACNT BYTE
                          ;------INITIALIZE CX FOR JUMP AFTER LAST PARAMETER IS PASSED TO NEC
EDDC  8A 46 0F                    MOV     AL,[BP+15]        ; RETRIEVE COMMAND PARAMETER
EDDF  A8 01                       TEST    AL,01H            ; IS THIS AN ODD NUMBERED FUNCTION?
EDE1  74 05                       JZ      J16_1             ; JUMP IF NOT ODD NUMBERED
EDE3  B9 EE4E R                   MOV     CX,OFFSET WRITE_LOOP
EDE6  EB 0C                       JMP     SHORT J16_3
EDE8  3C 02               J16_1:  CMP     AL,2              ; IS THIS A READ?
EDEA  75 05                       JNZ     J16_2             ; JUMP IF VERIFY
EDEC  B9 EE3A R                   MOV     CX,OFFSET READ_LOOP
EDEF  EB 03                       JMP     SHORT J16_3
EDF1  B9 EE20 R           J16_2:  MOV     CX,OFFSET VERIFY_LOOP
                          ;------FINISH INITIALIZATION
EDF4                      J16_3:
                          ;-------------------------------------------------------------
                          ;***NOTE***
                          ;ALL INTERRUPTS ARE ABOUT TO BE DISABLED.  THERE IS A POTENTIAL
                          ;       THAT THIS TIME PERIOD WILL BE LONG ENOUGH TO MISS TIME OF
                          ;       DAY INTERRUPTS.  FOR THIS REASON, TIMER1 WILL BE USED TO
                          ;       KEEP TRACK OF THE NUMBER OF TIME OF DAY INTERRUPTS WHICH
                          ;       WILL BE MISSED. THIS INFORMATION IS USED AFTER THE DISKETTE
                          ;       OPERATION TO UPDATE THE TIME OF DAY.
                          ;-------------------------------------------------------------
EDF4  B0 10                       MOV     AL,10H            ; DISABLE NMI
EDF6  E6 A0                       OUT     NMI_PORT,AL       ; NO KEYBOARD INTERRUPT
EDF8  E8 EB31 R                   CALL    CLOCK_WAIT        ; WAIT IF TIMER0 IS ABOUT TO
                                                            ; INTERRUPT
                          ;------ ENABLE WATCHDOG TIMER
                          ;-------------------------------------------------------------
                          ;***NOTE***
                          ;       GIVEN THE CURRENT SYSTEM CONFIGURATION A  METHOD IS NEEDED
                          ;       TO PULL THE NEC OUT OF "FATAL ERROR" SITUATIONS.  A TIMER
                          ;       ON THE ADAPTER CARD IS PROVIDED WHICH WILL PERFORM THIS
                          ;       FUNCTION. THE WATCHDOG TIMER ON THE ADAPTER CARD IS ENABLED
                          ;       AND STROBED BEFORE THE 8259 INTERRUPT 6 LINE IS ENABLED.
                          ;       THIS IS BECAUSE OF A GLITCH ON THE LINE LARGE ENOUGH TO
                          ;       TRIGGER AN INTERRUPT.
                          ;-------------------------------------------------------------
EDFB  E8 EB45 R                   CALL    GET_DRIVE         ; GET BIT MASK FOR DRIVE
EDFE  BA 00F2                     MOV     DX,NEC_CTL        ; CONTROL PORT TO NEC
EE01  0C E0                       OR      AL,FDC_RESET+WD_ENABLE+WD_STROBE
EE03  EE                          OUT     DX,AL             ; OUTPUT CONTROL INFO FOR
                                                            ; WATCHDOG(WD) ENABLE
EE04  24 A7                       AND     AL,FDC_RESET+WD_ENABLE+7H
EE06  EE                          OUT     DX,AL             ; OUTPUT CONTROL INFO TO STROBE
                                                            ; WATCHDOG
EE07  BA 00F4                     MOV     DX,NEC_STAT       ; PORT TO NEC STATUS
EE0A  B0 20                       MOV     AL,20H            ; SELECT TIMER1 INPUT FROM TIMER0
                                                            ; OUTPUT
EE0C  E6 A0                       OUT     NMI_PORT,AL
                          ;------ READ TIMER1 NOW AND SAVE THE INITIAL VALUE
EE0E  E8 E81A R                   CALL    READ_TIME         ; GET TIMER1 VALUE
EE11  89 46 12                    MOV     [BP+18],AX        ; SAVE INITIAL VALUE FOR CLOCK
                                                            ; UPDATE IN TEMPORAY STORAGE
EE14  E8 EAFC R                   CALL    DISABLE           ; DISABLE ALL INTERRUPTS
                          ;------ NEC BEGINS OPERATION WHEN NEC RECEIVES LAST PARAMETER
EE17  5B                          POP     BX                ; GET PARAMTER FROM STACK
EE18  E8 E9B4 R                   CALL    GET_PARM          ; OUTPUT LAST PARAMETER TO THE NEC
EE1B  58                          POP     AX                ; CAN NOW DISCARD THAT DUMMY RETURN
                                                            ; ADDRESS
EE1C  06                          PUSH    ES
EE1D  1F                          POP     DS                ; INITIALIZE DS FOR WRITE
EE1E  FF E1                       JMP     CX                ; JUMP TO APPROPRIATE R/W/V LOOP
                          ;-------------------------------------------------------------
                          ;***NOTE***
                          ;       DATA IS TRANSFERRED USING POLLING ALGORITHMS.  THESE LOOPS
                          ;       TRANSFER A DATA BYTE AT A TIME WHILE POLLING THE NEC FOR
                          ;       NEXT DATA BYTE AND COMPLETION STATUS.
                          ;-------------------------------------------------------------
                          ;-----VERIFY OPERATION
EE20                      VERIFY_LOOP:
EE20  EC                          IN      AL,DX             ; READ STATUS
EE21  A8 20                       TEST    AL,BUSY_BIT       ; HAS NEC ENTERED EXECUTION PHASE
                                                            ; YET?
EE23  74 FB                       JZ      VERIFY_LOOP       ; NO, CONTINUE SAMPLING
EE25                      J22_2:
EE25  A8 80                       TEST    AL,RQM            ; IS DATA READY?
EE27  75 07                       JNZ     J22_4             ; JUMP IF DATA TRANSFER IS READY
EE29  EC                          IN      AL,DX             ; READ STATUS PORT
EE2A  A8 20                       TEST    AL,BUSY_BIT       ; ARE WE DONE?
EE2C  75 F7                       JNZ     J22_2             ; JUMP IF MORE TRANSFERS
EE2E  EB 35                       JMP     SHORT OP_END      ; TRANSFER DONE
EE30  42                  J22_4:  INC     DX                ; POINT AT NEC DATA REGISTER
EE31  EC                          IN      AL,DX             ; READ DATA
EE32  4A                          DEC     DX                ; POINT AT NEC STATUS REGISTER
EE33  EC                          IN      AL,DX             ; READ STATUS PORT
EE34  A8 20                       TEST    AL,BUSY_BIT       ; ARE WE DONE?
EE36  75 ED                       JNZ     J22_2             ; CONTINUE
EE38  EB 2B                       JMP     SHORT OP_END      ; WE ARE DONE
```

```
                                   ;-----READ OPERATION
EE3A                               READ_LOOP:
EE3A  EC                                   IN      AL,DX             ; READ STATUS REGISTER
EE3B  A8 20                                TEST    AL,BUSY_BIT       ; HAS NEC STARTED THE EXECUTION
                                                                     ; PHASE?
EE3D  74 FB                                JZ      READ_LOOP         ; HAS NOT STATRED YET
EE3F  EC                           J22_5:  IN      AL,DX             ; READ STATUS PORT
EE40  A8 20                                TEST    AL,BUSY_BIT       ; HAS NEC COMPLETED EXECUTION
                                                                     ; PHASE?
EE42  74 21                                JZ      OP_END            ; JUMP IF EXECUTION PHASE IS OVER
EE44  A8 80                                TEST    AL,RQM            ; IS DATA READY?
EE46  74 F7                                JZ      J22_5             ; READ THE DATA
EE48  42                                   INC     DX                ; POINT AT NEC_DATA
EE49  EC                                   IN      AL,DX             ; READ DATA
EE4A  AA                                   STOSB                     ; TRANSFER DATA
EE4B  4A                                   DEC     DX                ; POINT AT NEC_STATUS
EE4C  EB F1                                JMP     J22_5             ; CONTINUE WITH READ OPERATION
                                   ;-----WRITE AND FORMAT OPERATION
EE4E                               WRITE_LOOP:
EE4E  EC                                   IN      AL,DX             ; READ NEC STATUS PORT
EE4F  A8 20                                TEST    AL,BUSY_BIT       ; HAS THE NEC ENTERED EXECUTION
                                                                     ; PHASE YET?
EE51  74 FB                                JZ      WRITE_LOOP        ; NO, CONTINUE LOOPING
EE53  B9 2080                              MOV     CX,BUSY_BIT*256+RQM
EE56                               J22_7:
EE56  EC                                   IN      AL,DX             ; READ STATUS PORT
EE57  84 C5                                TEST    AL,CH             ; IS THE FEC STILL IN THE EXECUTION
                                                                     ; PHASE?
EE59  74 0A                                JZ      OP_END            ; JUMP IF EXECUTION PHASE IS DONE.
EE5B  84 C1                                TEST    AL,CL             ; IS THE DATA PORT READY FOR THE
                                                                     ; TRANSFER?
EE5D  74 F7                                JZ      J22_7             ; JUMP TO WRITE DATA
EE5F  42                                   INC     DX                ; POINT AT DATA REGISTER
EE60  AC                                   LODSB                     ; TRANSFER BYTE
EE61  EE                                   OUT     DX,AL             ; WRITE THE BYTE ON THE DISKETTE
EE62  4A                                   DEC     DX                ; POINT AT THE STATUS REGISTER
EE63  EB F1                                JMP     J22_7             ; CONTINUE WITH WRITE OR FORMAT
                                   ;-------TRANSFER PROCESS IS OVER
EE65  9C                           OP_END: PUSHF                     ; SAVE THE CARRY BIT SET IN
                                                                     ; DISK_INT
EE66  E8 EB45 R                            CALL    GET_DRIVE         ; GET BIT MASK FOR DRIVE SELECTION
EE69  0C 80                                OR      AL,FDC_RESET      ; NO RESET, KEEP DRIVE SPINNING
EE6B  BA 00F2                              MOV     DX,NEC_CTL        ;
EE6E  EE                                   OUT     DX,AL             ; DISABLE WATCHDOG
                                   ;------ UPDATE TIME OF DAY
EE6F  E8 138B R                            CALL    DDS               ; POINT DS AT BIOS DATA SEGMENT
EE72  E8 EB31 R                            CALL    CLOCK_WAIT        ; WAIT IF TIMER0 IS CLOSE TO
                                                                     ; WRAPPING
EE75  E8 E81A R                            CALL    READ_TIME
EE78  8B 5E 12                             MOV     BX,[BP+18]        ; GET THE INITIAL VALUE OF TIMER1
EE7B  2B C3                                SUB     AX,BX             ; UPDATE NUMBER OF INTERRUPTS
                                                                     ; MISSED
EE7D  F7 D8                                NEG     AX                ; PUT IT IN AX
EE7F  50                                   PUSH    AX                ; SAVE IT FOR REUSE IN ISSUING USER
                                                                     ; TIMER INTERRUPTS
EE80  01 06 006C R                         ADD     TIMER_LOW,AX      ; ADD NUMBER OF TIMER INTERRUPTS TO
                                                                     ; TIME
EE84  73 04                                JNC     J16_4             ; JUMP IF TIMER_LOW DID NOT SPILL
                                                                     ; OVER TO TIMER_HI
EE86  FF 06 006E R                         INC     TIMER_HIGH
EE8A  83 3E 006E R 18              J16_4:  CMP     TIMER_HIGH,018H   ; TEST FOR COUNT TOTALING 24 HOURS
EE8F  75 19                                JNZ     J16_5             ; JUMP IF NOT 24 HOURS
EE91  81 3E 006C R 0080                    CMP     TIMER_LOW,080H    ; LOW VALUE = 24 HOUR VALUE?
EE97  7C 11                                JL      J16_5             ; NOT 24 HOUR VALUE?
                                   ;------ TIMER HAS GONE 24 HOURS
EE99  C7 06 006E R 0000                    MOV     TIMER_HIGH,0      ; ZERO OUT TIMER_HIGH VALUE
EE9F  81 2E 006C R 0080                    SUB     TIMER_LOW,080H    ; VALUE REFLECTS CORRECT TICKS PAST
                                                                     ; 0080H
EEA5  C6 06 0070 R 01                      MOV     TIMER_OFL,1       ; INDICATES 24 HOUR THRESHOLD
EEAA  E8 EB0B R                    J16_5:  CALL    ENABLE            ; ENABLE ALL INTERRUPTS
EEAD  59                                   POP     CX                ; CX:=AX, COUNT FOR NUMBER OF USER
                                                                     ; TIME INTERRUPTS
EEAE  E3 26                                JCXZ    J16_7             ; IF ZERO DO NOT ISSUE ANY
                                                                     ; INTERRUPTS
EEB0  1E                                   PUSH    DS                ; SAVE ALL REGISTERS SAVED PRIOR TO
                                                                     ; INT 1C CALL FROM TIMERINT
EEB1  50                                   PUSH    AX                ; THIS PROVIDES A COMPATIBLE
                                                                     ; INTERFACE TO 1C
EEB2  52                                   PUSH    DX                ;
EEB3                               J16_6:
EEB3  CD 1C                                INT     1CH               ; TRANSFER CONTROL TO USER
                                                                     ; INTERRUPT
EEB5  E2 FC                                LOOP    J16_6             ; DO ALL USER TIMER INTERRUPTS
EEB7  5A                                   POP     DX
EEB8  58                                   POP     AX
EEB9  1F                                   POP     DS                ; RESTORE REGISTERS
                                   ;------ CLOCK IS UPDATED AND USER INTERRUPTS 1C HAVE BEEN ISSUED.
                                   ;       CHECK IF KEYSTROKE OCCURED
EEBA  0A C0                                OR      AL,AL             ; AL WAS SET DURING CALL TO ENABLE
EEBC  74 18                                JZ      J16_7             ; NO KEY WAS PRESSED WHILE SYSTEM
                                                                     ; WAS MASKED
EEBE  BB 0080                              MOV     BX,080H           ; DURATION OF TONE
EEC1  B9 0048                              MOV     CX,048H           ; FREQUNCY OF TONE
EEC4  E8 E035 R                            CALL    KB_NOISE          ; NOTIFY USER OF MISSED KEYBORAD
                                                                     ; INPUT
```

```
                               ;-------CLEAR SHIFT STATES DONT LEAVE POSSIBILTY OF DANGLING STATES
                               ;       OF MISSED BREAKS
EEC7  80 26 0017 R F0                  AND     KB_FLAG,0F0H      ; CLEAR ALT,CLRL,LEFT AND RIGHT
                                                                 ; SHIFTS
EECC  80 26 0018 R 0F                  AND     KB_FLAG_1,0FH     ; CLEAR POTENTIAL BREAK OF INS,CAPS
                                                                 ; NUM AND SCROLL SHIFT
EED1  80 26 0088 R 1F                  AND     KB_FLAG_2,1FH     ; CLEAR FUNCTION STATES
EED6  9D                       J16_7:  POPF                      ; GET THE FLAGS
EED7                           J17:
EED7  72 40                            JC      J20
EED9  E8 EAA0 R                        CALL    RESULTS           ; GET THE NEC STATUS
EEDC  72 3B                            JC      J20               ; LOOK FOR ERROR
                               ;------ CHECK THE RESULTS RETURNED BY THE CONTROLLER
EEDE  FC                               CLD                       ; SET THE CORRECT DIRECTION
EEDF  BE 0042 R                        MOV     SI,OFFSET NEC_STATUS ; POINT TO STATUS FIELD
EEE2  AC                               LODS    NEC_STATUS        ; GET ST0
EEE3  24 C0                            AND     AL,0C0H           ; TEST FOR NORMAL TERMINATION
EEE5  74 58                            JZ      J22               ; OPN_OK
EEE7  3C 40                            CMP     AL,040H           ; TEST FOR ABNORMAL TERMINATION
EEE9  75 25                            JNZ     J18               ; NOT ABNORMAL, BAD NEC
                               ;-----------------------------------------------------------------
                               ;***NOTE***
                               ;       THE CURRENT SYSTEM CONFIGURATION HAS NO DMA.  IN ORDER TO
                               ;       STOP THE NEC AN EOT MUST BE PASSED TO FORCE THE NEC TO HALT
                               ;       THEREFORE, THE STATUS RETURNED BY THE NEC WILL ALWAYS SHOW
                               ;       AN EOT ERROR. IF THIS IS THE ONLY ERROR RETURNED AND THE
                               ;       NUMBER OF SECTORS TRANSFERRED EQUALS THE NUMBER SECTORS
                               ;       REQUESTED IN THIS INTERRUPT CALL THEN THE OPERATION HAS
                               ;       COMPLETED SUCCESSFULLY. IF AN EOT ERROR IS RETURNED AND THE
                               ;       REQUESTED NUMBER OF SECTORS IS NOT THE NUMBER OF SECTORS
                               ;       TRANSFERRED THEN THE ERROR IS LEGITIMATE. WHEN THE EOT
                               ;       ERROR IS INVALID THE STATUS BYTES RETURNED ARE UPDATED TO
                               ;       REFLECT THE STATUS OF THE OPERATION IF DMA HAD BEEN PRESENT
                               ;-----------------------------------------------------------------
EEEB  AC                               LODS    NEC_STATUS        ; GET ST1
EEEC  3C 80                            CMP     AL,80H            ; IS THIS THE ONLY ERROR?
EEEE  74 2A                            JE      J21_1             ; NORMAL TERMINATION, NO ERROR
EEF0  D0 E0                            SAL     AL,1              ; NOT EOT ERROR, BYPASS ERROR BITS
EEF2  D0 E0                            SAL     AL,1
EEF4  D0 E0                            SAL     AL,1              ; TEST FOR CRC ERROR
EEF6  B4 10                            MOV     AH,BAD_CRC
EEF8  72 18                            JC      J19               ; RW_FAIL
EEFA  D0 E0                            SAL     AL,1              ; TEST FOR DMA OVERRUN
EEFC  B4 08                            MOV     AH,BAD_DMA
EEFE  72 12                            JC      J19               ; RW_FAIL
EF00  D0 E0                            SAL     AL,1
EF02  D0 E0                            SAL     AL,1              ; TEST FOR RECORD NOT FOUND
EF04  B4 04                            MOV     AH,RECORD_NOT_FND
EF06  72 0A                            JC      J19               ; RW_FAIL
EF08  D0 E0                            SAL     AL,1
EF0A  D0 E0                            SAL     AL,1              ; TEST MISSING ADDRESS MARK
EF0C  B4 02                            MOV     AH,BAD_ADDR_MARK
EF0E  72 02                            JC      J19               ; RW_FAIL
                               ;------ NEC MUST HAVE FAILED
EF10                           J18:                              ; RW-NEC-FAIL
EF10  B4 20                            MOV     AH,BAD_NEC
EF12                           J19:                              ; RW-FAIL
EF12  08 26 0041 R                     OR      DISKETTE_STATUS,AH
EF16  E8 EAE1 R                        CALL    NUM_TRANS         ; HOW MANY WERE REALLY TRANSFERRED
EF19                           J20:                              ; RW_ERR
EF19  C3                               RET                       ; RETURN TO CALLER
                               ;------ OPERATION WAS SUCCESSFUL
EF1A                           J21_1:
EF1A  8A 5E 0E                         MOV     BL,[BP+14]        ; GET NUMBER OF SECTORS PASSED
                                                                 ; FROM STACK
EF1D  E8 EAE1 R                        CALL    NUM_TRANS         ; HOW MANY GOT MOVED, AL CONTAINS
                                                                 ; NUM OF SECTORS
EF20  3A D8                            CMP     BL,AL             ; NUMBER REQUESTED=NUMBER ACTUALLY
                                                                 ; TRANSFERRED?
EF22  74 0C                            JE      J21_2             ; TRANSFER SUCCESSFUL
                               ;------ OPERATION ATTEMPTED TO ACCESS DATA PAST REAL EOT.  THIS IS
                               ;       A REAL ERROR
EF24  80 0E 0041 R 04                  OR      DISKETTE_STATUS ,RECORD_NOT_FND
EF29  C6 06 0043 R 80                  MOV     NEC_STATUS+1,80H ; ST1 GETS CORRECT VALUE
EF2E  F9                               STC
EF2F  C3                               RET
EF30  33 C0                    J21_2:  XOR     AX,AX             ; CLEAR AX FOR NEC_STATUS UPDATE
EF32  33 F6                            XOR     SI,SI             ; INDEX TO NEC_STATUS ARRAY
EF34  88 84 0042 R                     MOV     NEC_STATUS[SI],AL ; ZERO OUT BYTE, ST0
EF38  46                               INC     SI                ; POINT INDEX AT SECOND BYTE
EF39  88 84 0042 R                     MOV     NEC_STATUS[SI],AL ; ZERO OUT BUYE, ST1
EF3D  EB 03                            JMP     SHORT J21_3       ; OPN_OK
EF3F  E8 EAE1 R                J22:    CALL    NUM_TRANS
EF42  32 E4                    J21_3:  XOR     AH,AH                     ; NO ERRORS
EF44  C3                               RET
EF45                           RW_OPN  ENDP
                               ;----------------------------------------------
                               ; DISK_INT
                               ;       THIS ROUTINE HANDLES THE DISKETTE INTERRUPT. AN INTERRUPT
                               ;       WILL OCCUR ONLY WHEN THE ONE-SHOT TIMER IS FIRED.  THIS
                               ;       OCCURS IN AN ERROR SITUATION.  THIS ROUTINE SETS ERRORS IN
                               ;       THE DISKETTE STATUS BYTE AND DISABLES THE ONE-SHOT TIMER.
                               ;       THEN THE RETURN ADDRESS ON THE STACK IS CHANGED TO RETURN
                               ;       TO THE OP_END LABEL.
                               ; INPUT
                               ;       NONE.
                               ; OUTPUT
                               ;       NONE.  DS POINTS AT BIOS DATA AREA.  CARRY FLAG IS SET SO
                               ;       THAT ERROR WILL BE CAUGHT IN THE ENVIRONMEMT RETURNED TO.
                               ;----------------------------------------------
```

```
EF57                                    ORG     0EF57H
EF57                            DISK_INT        PROC    FAR
EF57  1E                                PUSH    DS
EF58  50                                PUSH    AX
EF59  52                                PUSH    DX                ; SAVE REGISTER
EF5A  55                                PUSH    BP                ; SAVE THE BP REGISTER
EF5B  E8 1388 R                         CALL    DDS               ; SETUP DS TO POINT AT BIOS DATA
                                ;------ CHECK IF INTERRUPT OCCURED IN INT13 OR WHETHER IT IS A
                                ;       SPURIOUS INTERRUPT
EF5E  8B EC                             MOV     BP,SP             ; POINT BP AT STACK
EF60  0E                                PUSH    CS                ; WAS IT IN THE BIOS AREA
EF61  58                                POP     AX
EF62  3B 46 0A                          CMP     AX,WORD PTR[BP+10] ; GET INTERRUPTED SEGMENT
EF65  75 48                             JNE     DI3               ; NOT IN BIOS, ERROR CONDITION
EF67  8B 46 08                          MOV     AX,WORD PTR[BP+8] ; GET IP ON THE STACK
EF6A  3D EE20 R                         CMP     AX,OFFSET VERIFY_LOOP ; RANGE CHECK IP FOR DISK
                                                                  ; TRANSFER
EF6D  7C 40                             JL      DI3               ; BELOW TRANSFER CODE
EF6F  3D EE66 R                         CMP     AX,OFFSET OP_END+1 ; UPPER RANGE OF TRANSFER CODE
EF72  7D 3B                             JGE     DI3               ; ABOVE RANGE OF WATCHDOG TERRAIN
                                ;-------VALID DISKETTE INTERRUPT CHANGE RETURN ADDRESS ON STACK TO
                                ;       PULL OUT OF LOOP
EF74  C7 46 08 EE65 R                   MOV     WORD PTR[BP+8],OFFSET OP_END
EF79  81 4E 0C 0001                     OR      WORD PTR[BP+12],1 ; TURN ON CARRY FLAG IN FLAGS ON
                                                                  ; STACK
                                ;-----------------------------------------------------------------
                                ;***NOTE***
                                ; A WRITE PROTECTED DISKETTE WILL ALWAYS GET STUCK IN WRITE LOOP
                                ; WAITING FOR BEGINNING OF EXECUTION PHASE.  WHEN THE WATCHDOG
                                ; FIRES AND THE STATUS IN PORT NEC_STAT = DXH (X MEANS DON'T CARE)
                                ; STATUS FROM THE RESULT PHASE IS AVAILABLE.  THE STATUS IS READ
                                ; AND WRITE PROTECT IS  CHECKED FOR.
                                ;-----------------------------------------------------------------
EF7E  BA 00F4                           MOV     DX,NEC_STAT
EF81  EC                                IN      AL,DX             ; GET NEC STATUS BYTE
EF82  24 F0                             AND     AL,0F0H           ; MASK HIGH NIBBLE
EF84  3C D0                             CMP     AL,0D0H           ; IS EXECUTION PHASE DONE
EF86  75 14                             JNE     DI1               ; STUCK IN LOOP
EF88  E8 EAA0 R                         CALL    RESULTS           ; GET STATUS OF OPERATION
EF8B  BE 0042 R                         MOV     SI,OFFSET NEC_STATUS ; ADDRESS OF BYTES RETURNED BY
                                                                  ; NEC
EF8E  8A 44 01                          MOV     AL,[SI+1]         ; GET ST1
EF91  A8 02                             TEST    AL,02H            ; WRITE PROTECT SIGNAL ACTIVE?
EF93  74 07                             JZ      DI1               ; TIME OUT ERROR
EF95  80 0E 0041 R 03                   OR      DISKETTE_STATUS,WRITE_PROTECT
EF9A  EB 13                             JMP     SHORT DI3
                                ;------TIME OUT ERROR
EF9C  80 0E 0041 R 80           DI1:    OR      DISKETTE_STATUS,TIME_OUT
EFA1  C6 06 003E R 00                   MOV     SEEK_STATUS,0     ; SET RECAL ON DRIVES
                                ;------ RESET THE NEC AND DISABLE WATCHDOG
EFA6  BA 00F2                   DI2:    MOV     DX,NEC_CTL        ; ADDRESS TO NEC CONTROL PORT
EFA9  5D                                POP     BP                ; POINT BP AT BASE OF STACKED
                                                                  ; PARAMETERS
EFAA  E8 EB45 R                         CALL    GET_DRIVE         ; RESET ADAPTER AND DISABLE WD
EFAD  55                                PUSH    BP                ; RESTORE FOR RETURNED CALL
EFAE  EE                                OUT     DX,AL
EFAF  B0 20                     DI3:    MOV     AL,EOI            ; GIVE EOI TO 8259
EFB1  E6 20                             OUT     INTA00,AL
EFB3  5D                                POP     BP
EFB4  5A                                POP     DX
EFB5  58                                POP     AX
EFB6  1F                                POP     DS
EFB7  CF                                IRET                      ; RETURN FROM INTERRUPT
EFB8                            DISK_INT        ENDP
                                ;--------------------------------------------
                                ; DISK_BASE
                                ;  THIS IS THE SET OF PARAMETERS REQUIRED FOR
                                ;  DISKETTE OPERATION.  THEY ARE POINTED AT BY THE
                                ;  DATA VARIABLE DISK_POINTER.  TO MODIFY THE PARAMETERS,
                                ;  BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT
                                ;--------------------------------------------
EFC7                                    ORG     0EFC7H
EFC7                            DISK_BASE       LABEL   BYTE
EFC7  CF                                DB      11001111B         ; SRT=C, HD UNLOAD=0F - 1ST SPECIFY
                                                                  ; BYTE
EFC8  03                                DB      3                 ; HD LOAD=1, MODE=NO DMA - 2ND
                                                                  ; SPECIFY BYTE
EFC9  25                                DB      MOTOR_WAIT        ; WAIT AFTER OPN TIL MOTOR OFF
EFCA  02                                DB      2                 ; 512 BYTES/SECTOR
EFCB  08                                DB      8                 ; EOT ( LAST SECTOR ON TRACK)
EFCC  2A                                DB      02AH              ; GAP LENGTH
EFCD  FF                                DB      0FFH              ; DTL
EFCE  50                                DB      050H              ; GAP LENGTH FOR FORMAT
EFCF  F6                                DB      0F6H              ; FILL BYTE FOR FORMAT
EFD0  19                                DB      25                ; HEAD SETTLE TIME (MILLISECONDS)
EFD1  04                                DB      4                 ; MOTOR START TIME (1/8 SECONDS)
```

```
                               ;--- INT 17 -----------------------------------------------
                               ; PRINTER_IO
                               ;       THIS ROUTINE PROVIDES COMMUNICATION WITH THE PRINTER
                               ;       (AH)=0  PRINT THE CHARACTER IN (AL)
                               ;               ON RETURN, AH=1 IF CHARACTER COULD NOT BE PRINTED
                               ;               (TIME OUT), OTHER BITS SET AS ON NORMAL STATUS CALL
                               ;       (AH)=1  INITIALIZE THE PRINTER PORT
                               ;               RETURNS WITH (AH) SET WITH PRINTER STATUS
                               ;       (AH)=2  READ THE PRINTER STATUS INTO (AH)
                               ;       7       6       5       4       3       2-1     0
                               ;       :       :       :       :       :       :       :_ TIME OUT
                               ;       :       :       :       :       :       :_ UNUSED
                               ;       :       :       :       :       :_ 1 = I/O ERROR
                               ;       :       :       :       :_ 1 = SELECTED
                               ;       :       :       :_ 1 = OUT OF PAPER
                               ;       :       :_ 1 = ACKNOWLEDGE
                               ;       :_ 1 = NOT BUSY
                               ;
                               ;       (DX) = PRINTER TO BE USED (0,1,2) CORRESPONDING TO ACTUAL
                               ;               VALUES IN PRINTER_BASE AREA
                               ; DATA AREA PRINTER_BASE CONTAINS THE BASE ADDRESS OF THE PRINTER
                               ; CARD(S) AVAILABLE (LOCATED AT BEGINNING OF DATA SEGMENT, 408H
                               ; ABSOLUTE, 3 WORDS), UNLESS THERE IS ONLY A SERIAL PRINTER
                               ; ATTACHED, IN WHICH CASE THE WORD AT 40:8 WILL CONTAIN A 02F8H.
                               ; REGISTERS     AH IS MODIFIED
                               ;               ALL OTHERS UNCHANGED
                               ;-----------------------------------------------------------------
                                       ASSUME  CS:CODE,DS:DATA
EFD2                                   ORG     0EFD2H
EFD2                           PRINTER_IO      PROC    FAR
EFD2  FB                               STI                     ; INTERRUPTS BACK ON
EFD3  1E                               PUSH    DS              ; SAVE SEGMENT
EFD4  52                               PUSH    DX
EFD5  56                               PUSH    SI
EFD6  51                               PUSH    CX
EFD7  53                               PUSH    BX
EFD8  E8 138B R                        CALL    DDS
                               ;REDIRECT TO SERIAL ONLY IF:
                               ; 1> SERIAL PRINTER IS ATTACHED,AND...
                               ; 2> WORD AT PRINTER BASE = 02F8H.
                               ; POWER ONS WILL ONLY PUT A 02F8H IN THE PRINTER BASE IF THERE'S
                               ; NO PARALLEL PRINTER ATTACHED.
EFDB  8B 0E 0010 R                     MOV     CX,EQUIP_FLAG   ;GET FLAG IN CX
EFDF  F6 C5 20                         TEST    CH,00100000B    ;SERIAL ATTACHED?
EFE2  74 0D                            JZ      B0              ;NO -HANDLE NORMALLY
EFE4  8B 1E 0008 R                     MOV     BX,PRINTER_BASE ;SEE IF THERE'S AN RS232
EFE8  81 FB 02F8                       CMP     BX,02F8H        ;BASE IN THE PRINTER BASE.
EFEC  75 03                            JNE     B0
EFEE  E9 18C3 R                B00:    JMP     B1_A            ;IF THERE IS REDIRECT
                                                               ;ELSE... HANDLE AS PARALLEL
                               ;CONTROL IS PASSED TO THIS POINT IF THERE IS A PARALLEL OR
                               ;THERE'S NO SERIAL PRINTER ATTACHED.
EFF1  8B F2                    B0:     MOV     SI,DX           ; GET PRINTER PARM
EFF3  8A 9C 0078 R                     MOV     BL,PRINT_TIM_OUT[SI] ; LOAD TIMEOUT VALUE
EFF7  D1 E6                            SHL     SI,1            ; WORD OFFSET INTO TABLE
EFF9  8B 94 0008 R                     MOV     DX,PRINTER_BASE[SI] ; GET BASE ADDRESS FOR PRINTER
                                                               ; CARD
EFFD  0B D2                            OR      DX,DX           ; TEST DX FOR ZERO, INDICATING NO
                                                               ; PRINTER
EFFF  74 0C                            JZ      B1              ; IF NO PARALLEL, RETURN
F001  0A E4                            OR      AH,AH           ; TEST FOR (AH)=0
F003  74 0E                            JZ      B2              ; PRINT_AL
F005  FE CC                            DEC     AH              ; TEST FOR (AH)=1
F007  74 40                            JZ      B8              ; INIT_PRT
F009  FE CC                            DEC     AH              ; TEST FOR (AH)=2
F00B  74 28                            JZ      B5              ; PRINTER STATUS
F00D                           B1:                             ; RETURN
F00D  5B                               POP     BX
F00E  59                               POP     CX
F00F  5E                               POP     SI              ; RECOVER REGISTERS
F010  5A                               POP     DX              ; RECOVER REGISTERS
F011  1F                               POP     DS
F012  CF                               IRET
                               ;------ PRINT THE CHARACTER IN (AL)
F013  50                       B2:     PUSH    AX              ; SAVE VALUE TO PRINT
F014  EE                               OUT     DX,AL           ; OUTPUT CHAR TO PORT
F015  42                               INC     DX              ; POINT TO STATUS PORT
                               ;
                               ;------WAIT BUSY
                               ;
F016  2B C9                    B3:     SUB     CX,CX           ; INNER LOOP (64K)
F018  EC                       B3_1:   IN      AL,DX           ; GET STATUS
F019  8A E0                            MOV     AH,AL           ; STATUS TO AH ALSO
F01B  A8 80                            TEST    AL,80H          ; IS THE PRINTER CURRENTLY BUSY
F01D  75 0E                            JNZ     B4              ; OUT_STROBE
F01F  E2 F7                            LOOP    B3_1            ; LOOP IF NOT
F021  FE CB                            DEC     BL              ; DROP OUTER LOOP COUNT
F023  75 F1                            JNZ     B3              ; MAKE ANOTHER PASS IF NOT ZERO
F025  80 CC 01                         OR      AH,1            ; SET ERROR FLAG
F028  80 E4 F9                         AND     AH,0F9H         ; TURN OFF THE UNUSED BITS
F02B  EB 14                            JMP     SHORT B7        ; RETURN WITH ERROR FLAG SET
F02D                           B4:                             ; OUT_STROBE
F02D  B0 0D                            MOV     AL,0DH          ; SET THE STROBE HIGH
F02F  42                               INC     DX
F030  EE                               OUT     DX,AL
F031  B0 0C                            MOV     AL,0CH          ; SET THE STROBE LOW
F033  EE                               OUT     DX,AL
F034  58                               POP     AX              ; RECOVER THE OUTPUT CHAR
```

```
                                ;------ PRINTER STATUS
F035  50                        B5:     PUSH    AX              ; SAVE AL REG
F036  8B 94 0008 R              B6:     MOV     DX,PRINTER_BASE[SI]
F03A  42                                INC     DX
F03B  EC                                IN      AL,DX           ; GET PRINTER STATUS
F03C  8A E0                             MOV     AH,AL
F03E  80 E4 F8                          AND     AH,0F8H         ; TURN OFF UNUSED BITS
F041                            B7:                             ; STATUS_SET
F041  5A                                POP     DX              ; RECOVER AL REG
F042  8A C2                             MOV     AL,DL           ; GET CHARACTER INTO AL
F044  80 F4 48                          XOR     AH,48H          ; FLIP A COUPLE OF BITS
F047  EB C4                             JMP     B1              ; RETURN FROM ROUTINE
                                ;------ INITIALIZE THE PRINTER PORT
F049  50                        B8:     PUSH    AX              ; SAVE AL
F04A  42                                INC     DX              ; POINT TO OUTPUT PORT
F04B  42                                INC     DX
F04C  B0 08                             MOV     AL,8            ; SET INIT LINE LOW
F04E  EE                                OUT     DX,AL
F04F  B8 03E8                           MOV     AX,1000
F052                            B9:                             ; INIT_LOOP
F052  48                                DEC     AX              ; LOOP FOR RESET TO TAKE
F053  75 FD                             JNZ     B9              ; INIT_LOOP
F055  B0 0C                             MOV     AL,0CH          ; NO INTERRUPTS, NON AUTO LF, INIT
                                                                ; HIGH
F057  EE                                OUT     DX,AL
F058  EB DC                             JMP     B6              ; PRT_STATUS_1
F05A                            PRINTER_IO      ENDP
F065                                    ORG     0F065H
F065  E9 0D08 R                         JMP     NEAR PTR VIDEO_IO
                                ;--------------------------------------------
                                ;    SUBROUTINE TO SAVE  ANY SCAN CODE RECEIVED ;
                                ;    BY THE NMI ROUTINE (PASSED IN AL)          ;
                                ;    DURING POST IN THE KEYBOARD BUFFER         ;
                                ;    CALLED THROUGH INT. 48H                    ;
                                ;--------------------------------------------
F068                            KEY_SCAN_SAVE   PROC    FAR
                                        ASSUME  DS:DATA
F068  E8 138B R                         CALL    DDS             ; POINT DS TO DATA AREA
F06B  BE 001E R                         MOV     SI,OFFSET KB_BUFFER ; POINT TO FIRST LOC. IN BUFFER
F06E  88 04                             MOV     [SI],AL         ; SAVE SCAN CODE
F070  8B C4                             MOV     AX,SP           ; CHECK FOR STACK UNDERFLOW
F072  80 E4 E0                          AND     AH,11100000B    ; (THESE BITS WILL BE 111 IF
                                                                ; UNDERFLOW HAPPEND)
F075  74 0D                             JZ      KS_1
F077  32 C0                             XOR     AL,AL
F079  E6 A0                             OUT     0A0H,AL         ; SHUT OFF NMI
F07B  BB 2000                           MOV     BX,2000H        ; ERROR CODE 2000H
F07E  BE 0036 R                         MOV     SI,OFFSET KEY_ERR ; POST MESSAGE
F081  E8 09BC R                         CALL    E_MSG           ; AND HALT SYSTEM
F084  CF                        KS_1:   IRET                    ; RETURN TO CALLER
F085                            KEY_SCAN_SAVE   ENDP
                                ;------------------------------------------------------------
                                ;   SUBROUTINE TO SET AN INS8250 CHIP'S BAUD RATE TO 9600 BPS AND
                                ;   DEFINE IT'S DATA WORD AS HAVING 8 BITS/WORD, 2 STOP BITS, AND
                                ;   ODD PARITY.
                                ;
                                ;   EXPECTS TO BE PASSED:
                                ;       (DX) = LINE CONTROL REGISTER
                                ;
                                ;   UPON RETURN:
                                ;       (DX) = TRANSMIT/RECEIVE BUFFER ADDRESS
                                ;
                                ;   ALSO, ALTERS REGISTER AL.  ALL OTHERS REMAIN INTACT.
                                ;------------------------------------------------------------
F085                            S8250   PROC    NEAR
F085  B0 80                             MOV     AL,80H          ; SET DLAB = 1
F087  EE                                OUT     DX,AL
F088  EB 00                             JMP     $+2             ; I/O DELAY
F08A  83 EA 03                          SUB     DX,3            ; LSB OF DIVISOR LATCH
F08D  B0 0C                             MOV     AL,12           ; DIVISOR = 12 PRODUCES 9600 BPS
F08F  EE                                OUT     DX,AL           ; SET LSB
F090  EB 00                             JMP     $+2             ; I/O DELAY
F092  42                                INC     DX              ; MSB OF DIVISOR LATCH
F093  B0 00                             MOV     AL,0            ; HIGH ORDER OF DIVISORS
F095  EE                                OUT     DX,AL           ; SET MSB
F096  EB 00                             JMP     $+2             ; I/O DELAY
F098  42                                INC     DX
F099  42                                INC     DX              ; LINE CONTROL REGISTER
F09A  B0 0F                             MOV     AL,00001111B    ; 8 BITS/WORD, 2 STOP BITS, ODD
                                                                ; PARITY
F09C  EE                                OUT     DX,AL
F09D  EB 00                             JMP     $+2             ; I/O DELAY
F09F  83 EA 03                          SUB     DX,3            ; RECEIVER BUFFER
F0A2  EC                                IN      AL,DX           ; IN CASE WRITING TO PORT LCR
                                                                ; CAUSED DATA READY TO GO HIGH!
F0A3  C3                                RET
F0A4                            S8250   ENDP
                                ;------ TABLES FOR USE IN SETTING OF CRT MODE
F0A4                                    ORG     0F0A4H
F0A4                            VIDEO_PARMS     LABEL   BYTE
                                ;------ INIT_TABLE
F0A4  38 28 2C 06 1F 06                 DB      38H,28H,2CH,06H,1FH,6,19H ; SETUP FOR 40X25
      19
F0AB  1C 02 07 06 07                    DB      1CH,2,7,6,7
F0B0  00 00 00 00                       DB      0,0,0,0
```

```
= 0010                          M0040   EQU     $-VIDEO_PARMS

F0B4  71 50 5A 0C 1F 06                 DB      71H,50H,5AH,0CH,1FH,6,19H ; SETUP FOR 80X25
      19
F0BB  1C 02 07 06 07                    DB      1CH,2,7,6,7
F0C0  00 00 00 00                       DB      0,0,0,0

F0C4  38 28 2B 06 7F 06                 DB      38H,28H,2BH,06H,7FH,6,64H ; SET UP FOR GRAPHICS
      64
F0CB  70 02 01 26 07                    DB      70H,2,1,26H,7
F0D0  00 00 00 00                       DB      0,0,0,0

F0D4  71 50 56 0C 3F 06                 DB      71H,50H,56H,0CH,3FH,6,32H ; SET UP FOR GRAPHICS
      32
F0DB  38 02 03 26 07                    DB      38H,2,3,26H,7             ; USING 32K OF MEMORY
F0E0  00 00 00 00                       DB      0,0,0,0                   ; (MODES 9 & A)
                                ;-----------------------------------------
                                ; READ_AC_CURRENT
                                ;       THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE
                                ;       CURRENT CURSOR POSITION AND RETURNS THEM TO THE CALLER
                                ; INPUT
                                ;       (AH) = CURRENT CRT MODE
                                ;       (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )
                                ;       (DS) = DATA SEGMENT
                                ;       (ES) = REGEN SEGMENT
                                ; OUTPUT
                                ;       (AL) = CHAR READ
                                ;       (AH) = ATTRIBUTE READ
                                ;-----------------------------------------
                                        ASSUME  CS:CODE,DS:DATA,ES:DATA
F0E4                            READ_AC_CURRENT PROC    NEAR
F0E4  80 FC 04                          CMP     AH,4            ; IS THIS GRAPHICS?
F0E7  72 03                             JC      C60
F0E9  E9 F531 R                         JMP     GRAPHICS_READ
F0EC                            C60:                            ; READ_AC_CONTINUE
F0EC  E8 F0F7 R                         CALL    FIND_POSITION
F0EF  8B F3                             MOV     SI,BX           ; ESTABLISH ADDRESSING IN SI
F0F1  06                                PUSH    ES              ;
F0F2  1F                                POP     DS              ; GET SEGMENT FOR QUICK ACCESS
F0F3  AD                                LODSW                   ; GET THE CHAR/ATTR
F0F4  E9 0F70 R                         JMP     VIDEO_RETURN
F0F7                            READ_AC_CURRENT ENDP
F0F7                            FIND_POSITION   PROC    NEAR
F0F7  8A CF                             MOV     CL,BH           ; DISPLAY PAGE TO CX
F0F9  32 ED                             XOR     CH,CH
F0FB  8B F1                             MOV     SI,CX           ; MOVE TO SI FOR INDEX
F0FD  D1 E6                             SAL     SI,1            ; * 2 FOR WORD OFFSET
F0FF  8B 84 0050 R                      MOV     AX,[SI+ OFFSET CURSOR_POSN] ; GET ROW/COLUMN OF
                                                                ; THAT PAGE
F103  33 DB                             XOR     BX,BX           ; SET START ADDRESS TO ZERO
F105  E3 06                             JCXZ    C62             ; NO_PAGE
F107                            C61:                            ; PAGE_LOOP
F107  03 1E 004C R                      ADD     BX,CRT_LEN      ; LENGTH OF BUFFER
F10B  E2 FA                             LOOP    C61
F10D                            C62:                            ; NO_PAGE
F10D  E8 E5C2 R                         CALL    POSITION        ; DETERMINE LOCATION IN REGEN
F110  03 D8                             ADD     BX,AX           ; ADD TO START OF REGEN
F112  C3                                RET
F113                            FIND_POSITION   ENDP
                                ;-----------------------------------------
                                ; WRITE_AC_CURRENT
                                ;       THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER AT
                                ;       THE CURRENT CURSOR POSITION
                                ; INPUT
                                ;       (AH) = CURRENT CRT MODE
                                ;       (BH) = DISPLAY PAGE
                                ;       (CX) = COUNT OF CHARACTERS TO WRITE
                                ;       (AL) = CHAR TO WRITE
                                ;       (BL) = ATTRIBUTE OF CHAR TO WRITE
                                ;       (DS) = DATA SEGMENT
                                ;       (ES) = REGEN SEGMENT
                                ; OUTPUT
                                ;       NONE
                                ;-------------------------------------------
F113                            WRITE_AC_CURRENT        PROC    NEAR
F113  80 FC 04                          CMP     AH,4            ; IS THIS GRAPHICS?
F116  72 03                             JC      C63
F118  E9 F3F1 R                         JMP     GRAPHICS_WRITE
F11B                            C63:                            ; WRITE_AC_CONTINUE
F11B  8A E3                             MOV     AH,BL           ; GET ATTRIBUTE TO AH
F11D  50                                PUSH    AX              ; SAVE ON STACK
F11E  51                                PUSH    CX              ; SAVE WRITE COUNT
F11F  E8 F0F7 R                         CALL    FIND_POSITION
F122  8B FB                             MOV     DI,BX           ; ADDRESS TO DI REGISTER
F124  59                                POP     CX              ; WRITE COUNT
F125  58                                POP     AX              ; CHARACTER IN AX REG
F126                            C64:                            ; WRITE_LOOP
F126  AB                                STOSW                   ; PUT THE CHAR/ATTR
F127  E2 FD                             LOOP    C64             ;  AS MANY TIMES AS REQUESTED
F129  E9 0F70 R                         JMP     VIDEO_RETURN
F12C                            WRITE_AC_CURRENT        ENDP
```

```
                               ;----------------------------------------
                               ; WRITE_C_CURRENT
                               ;        THIS ROUTINE WRITES THE CHARACTER AT
                               ;        THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED
                               ; INPUT
                               ;        (AH) = CURRENT CRT MODE
                               ;        (BH) = DISPLAY PAGE
                               ;        (CX) = COUNT OF CHARACTERS TO WRITE
                               ;        (AL) = CHAR TO WRITE
                               ;        (DS) = DATA SEGMENT
                               ;        (ES) = REGEN SEGMENT
                               ; OUTPUT
                               ;        NONE
                               ;-------------------------------------------
F12C                           WRITE_C_CURRENT PROC    NEAR
F12C  80 FC 04                         CMP     AH,4            ; IS THIS GRAPHICS?
F12F  72 03                            JC      C65
F131  E9 F3F1 R                        JMP     GRAPHICS_WRITE
F134  50                       C65:    PUSH    AX              ; SAVE ON STACK
F135  51                               PUSH    CX              ; SAVE WRITE COUNT
F136  E8 F0F7 R                        CALL    FIND_POSITION
F139  8B FB                            MOV     DI,BX           ; ADDRESS TO DI
F13B  59                               POP     CX              ; WRITE COUNT
F13C  5B                               POP     BX              ; BL HAS CHAR TO WRITE
F13D                           C66:                            ; WRITE_LOOP
F13D  8A C3                            MOV     AL,BL           ; RECOVER CHAR
F13F  AA                               STOSB                   ; PUT THE CHAR/ATTR
F140  47                               INC     DI              ; BUMP POINTER PAST ATTRIBUTE
F141  E2 FA                            LOOP    C66             ;  AS MANY TIMES AS REQUESTED
F143  E9 0F70 R                        JMP     VIDEO_RETURN
F146                           WRITE_C_CURRENT ENDP
                               ;-------------------------------------------
                               ; READ DOT  -- WRITE DOT
                               ; THESE ROUTINES WILL WRITE A DOT, OR READ THE
                               ;  DOT AT THE INDICATED LOCATION
                               ; ENTRY --
                               ;   DX = ROW (0-199)     (THE ACTUAL VALUE DEPENDS ON THE MODE)
                               ;   CX = COLUMN ( 0-639) ( THE VALUES ARE NOT RANGE CHECKED )
                               ;   AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
                               ;        REQ'D FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
                               ;        BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
                               ;   DS = DATA SEGMENT
                               ;   ES = REGEN SEGMENT
                               ;
                               ; EXIT
                               ;        AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
                               ;--------------------------------------------
                                       ASSUME  CS:CODE,DS:DATA,ES:DATA
F146                           READ_DOT        PROC    NEAR
F146  80 3E 0049 R 0A                  CMP     CRT_MODE,0AH    ; 640X200 4 COLOR?
F14B  74 11                            JE      READ_ODD        ; YES, HANDLE SEPARATELY
F14D  E8 F1D9 R                        CALL    C72             ; DETERMINE BYTE POSITION OF DOT
F150  26: 8A 04                        MOV     AL,ES:[SI]      ; GET THE BYTE
F153  22 C4                            AND     AL,AH           ; MASK OFF THE OTHER BITS IN THE
                                                               ; BYTE
F155  D2 E0                            SHL     AL,CL           ; LEFT JUSTIFY THE VALUE
F157  8A CE                            MOV     CL,DH           ; GET NUMBER OF BITS IN RESULT
F159  D2 C0                            ROL     AL,CL           ; RIGHT JUSTIFY THE RESULT
F15B  E9 0F70 R                        JMP     VIDEO_RETURN    ; RETURN FROM VIDEO IO
                               ; IN 640X200 4 COLOR MODE, THE 2 COLOR BITS (C1,C0) ARE DIFFERENT
                               ; THAN OTHER MODES. C0 IS IN THE EVEN BYTE, C1 IS IN THE FOLLOWING
                               ; ODD BYTE - BOTH AT THE SAME BIT POSITION WITHIN THEIR RESPECTIVE
                               ; BYTES.
F15E                           READ_ODD:
F15E  E8 F1D9 R                        CALL    C72             ; DETERMINE POSITION OF DOT
F161  52                               PUSH    DX              ; SAVE INFO
F162  51                               PUSH    CX
F163  50                               PUSH    AX
F164  26: 8A 44 01                     MOV     AL,ES:[SI+1]    ; GET C1 COLOR BIT FROM ODD BYTE
F168  22 C4                            AND     AL,AH           ; MASK OFF OTHER BITS
F16A  D2 E0                            SHL     AL,CL           ; LEFT JUSTIFY THE VALUE
F16C  8A CE                            MOV     CL,DH           ; GET NUMBER OF BITS IN RESULT
F16E  FE C1                            INC     CL
F170  D2 C0                            ROL     AL,CL           ; RIGHT JUSTIFY THE RESULT
F172  8B D8                            MOV     BX,AX           ; SAVE IN BX REG
F174  58                               POP     AX              ; RESTORE POSITION INFO
F175  59                               POP     CX              ;
F176  5A                               POP     DX
F177  26: 8A 04                        MOV     AL,ES:[SI]      ; GET C0 COLOR BIT FROM EVEN BYTE
F17A  22 C4                            AND     AL,AH           ; MASK OFF OTHER BITS
F17C  D2 E0                            SHL     AL,CL           ; LEFT JUSTIFY THE VALUE
F17E  8A CE                            MOV     CL,DH           ; GET NUMBER OF BITS IN RESULT
F180  D2 C0                            ROL     AL,CL           ; RIGHT JUSTIFY THE RESULT
F182  0A C3                            OR      AL,BL           ; COMBINE C1 & C0
F184  E9 0F70 R                        JMP     VIDEO_RETURN
```

```
F187                          READ_DOT        ENDP
F187                          WRITE_DOT       PROC    NEAR
F187  51                              PUSH    CX                  ; SAVE COL
F188  52                              PUSH    DX                  ; SAVE ROW
F189  50                              PUSH    AX                  ; SAVE DOT VALUE
F18A  50                              PUSH    AX                  ;  TWICE
F18B  E8 F1D9 R                       CALL    C72                 ; DETERMINE BYTE POSITION OF THE
                                                                  ; DOT
F18E  D2 E8                           SHR     AL,CL               ; SHIFT TO SET UP THE BITS FOR
                                                                  ; OUTPUT
F190  22 C4                           AND     AL,AH               ; STRIP OFF THE OTHER BITS
F192  26: 8A 0C                       MOV     CL,ES:[SI]          ; GET THE CURRENT BYTE
F195  5B                              POP     BX                  ; RECOVER XOR FLAG
F196  F6 C3 80                        TEST    BL,80H              ; IS IT ON
F199  75 36                           JNZ     C70                 ; YES, XOR THE DOT
F19B  F6 D4                           NOT     AH                  ; SET THE MASK TO REMOVE THE
                                                                  ; INDICATED BITS
F19D  22 CC                           AND     CL,AH
F19F  0A C1                           OR      AL,CL               ; OR IN THE NEW VALUE OF THOSE BITS
F1A1                          C67:                                ; FINISH_DOT
F1A1  26: 88 04                       MOV     ES:[SI],AL          ; RESTORE THE BYTE IN MEMORY
F1A4  58                              POP     AX
F1A5  5A                              POP     DX                  ; RECOVER ROW
F1A6  59                              POP     CX                  ; RECOVER COL
F1A7  80 3E 0049 R 0A                 CMP     CRT_MODE,0AH        ; 640X200 4 COLOR?
F1AC  75 20                           JNE     C69                 ; NO,JUMP
F1AE  50                              PUSH    AX                  ; SAVE DOT VALUE
F1AF  50                              PUSH    AX                  ;  TWICE
F1B0  D0 E8                           SHR     AL,1                ; SHIFT c1 BIT INTO c0 POSITION
F1B2  E8 F1D9 R                       CALL    C72                 ; DETERMINE BYTE POSITION OF THE
                                                                  ; DOT
F1B5  D2 E8                           SHR     AL,CL               ; SHIFT TO SET UP THE BITS FOR
                                                                  ; OUTPUT
F1B7  22 C4                           AND     AL,AH               ; STRIP OFF THE OTHER BITS
F1B9  26: 8A 4C 01                    MOV     CL,ES:[SI+1]        ; GET THE CURRENT BYTE
F1BD  5B                              POP     BX                  ; RECOVER XOR FLAG
F1BE  F6 C3 80                        TEST    BL,80H              ; IS IT ON
F1C1  75 12                           JNZ     C71                 ; YES, XOR THE DOT
F1C3  F6 D4                           NOT     AH                  ; SET THE MASK TO REMOVE THE
                                                                  ; INDICATED BITS
F1C5  22 CC                           AND     CL,AH
F1C7  0A C1                           OR      AL,CL               ; OR IN THE NEW VALUE OF THOSE BITS
F1C9                          C68:                                ; FINISH_DOT
F1C9  26: 88 44 01                    MOV     ES:[SI+1],AL        ; RESTORE THE BYTE IN MEMORY
F1CD  58                              POP     AX
F1CE  E9 0F70 R               C69:    JMP     VIDEO_RETURN        ; RETURN FROM VIDEO IO
F1D1                          C70:                                ; XOR_DOT
F1D1  32 C1                           XOR     AL,CL               ; EXCLUSIVE OR THE DOTS
F1D3  EB CC                           JMP     C67                 ; FINISH UP THE WRITING
F1D5                          C71:                                ; XOR_DOT
F1D5  32 C1                           XOR     AL,CL               ; EXCLUSIVE OR THE DOTS
F1D7  EB F0                           JMP     C68                 ; FINISH UP THE WRITING
F1D9                          WRITE_DOT       ENDP
                              ;--------------------------------------------
                              ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
                              ; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
                              ; ENTRY --
                              ;  DX = ROW VALUE (0-199)
                              ;  CX = COLUMN VALUE (0-639)
                              ; EXIT --
                              ;  SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
                              ;  AH = MASK TO STRIP OFF THE BITS OF INTEREST
                              ;  CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
                              ;  DH = # BITS IN RESULT
                              ;---------------------------------------------
F1D9                          C72     PROC    NEAR
F1D9  53                              PUSH    BX                  ; SAVE BX DURING OPERATION
F1DA  50                              PUSH    AX                  ; WILL SAVE AL DURING OPERATION
                              ;------ DETERMINE 1ST BYTE IN IDICATED ROW BY MULTIPLYING ROW VALUE
                              ;       BY 40( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW
F1DB  B0 28                           MOV     AL,40
F1DD  52                              PUSH    DX                  ; SAVE ROW VALUE
F1DE  80 E2 FE                        AND     DL,0FEH             ; STRIP OFF ODD/EVEN BIT
F1E1  80 3E 0049 R 09                 CMP     CRT_MODE,09H        ; MODE USING 32K REGEN?
F1E6  72 03                           JC      C73                 ; NO, JUMP
F1E8  80 E2 FC                        AND     DL,0FCH             ; STRIP OFF LOW 2 BITS
F1EB  F6 E2                   C73:    MUL     DL                  ; AX HAS ADDRESS OF 1ST BYTE OF
                                                                  ; INDICATED ROW
F1ED  5A                              POP     DX                  ; RECOVER IT
F1EE  F6 C2 01                        TEST    DL,1                ; TEST FOR EVEN/ODD
F1F1  74 03                           JZ      C74                 ; JUMP IF EVEN ROW
F1F3  05 2000                         ADD     AX,2000H            ; OFFSET TO LOCATION OF ODD ROWS
F1F6                          C74:                                ; EVEN_ROW
F1F6  80 3E 0049 R 09                 CMP     CRT_MODE,09H        ; MODE USING 32K REGEN?
F1FB  72 08                           JC      C75                 ; NO, JUMP
F1FD  F6 C2 02                        TEST    DL,2                ; TEST FOR ROW 2 OR ROW 3
F200  74 03                           JZ      C75                 ; JUMP IF ROW 0 OR 1
F202  05 4000                         ADD     AX,4000H            ; OFFSET TO LOCATION OF ROW 2 OR 3
F205  8B F0                   C75:    MOV     SI,AX               ; MOVE POINTER TO SI
F207  58                              POP     AX                  ; RECOVER AL VALUE
F208  8B D1                           MOV     DX,CX               ; COLUMN VALUE TO DX
```

```
                                    ;------ DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
                                    ;SET UP THE REGISTERS ACCORDING TO THE MODE
                                    ;CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3/1 FOR HIGH/MED/LOW RES)
                                    ;CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2/1 FOR H/M/L)
                                    ;BL = MASK TO SELECT BITS FROM POINTED BYTE (80H/C0H/F0H FOR H/M/L)
                                    ;BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2/4 FOR H/M/L)
F20A  BB 02C0                               MOV     BX,2C0H
F20D  B9 0302                               MOV     CX,302H         ; SET PARMS FOR MED RES
F210  80 3E 0049 R 04                       CMP     CRT_MODE,4
F215  74 21                                 JE      C77             ; HANDLE IF MED RES
F217  80 3E 0049 R 05                       CMP     CRT_MODE,5
F21C  74 1A                                 JE      C77             ; HANDLE IF MED RES
F21E  BB 04F0                               MOV     BX,4F0H
F221  B9 0101                               MOV     CX,101H         ;SET PARMS FOR LOW RES
F224  80 3E 0049 R 0A                       CMP     CRT_MODE,0AH
F229  74 07                                 JE      C76             ; HANDLE MODE A AS HIGH RES
F22B  80 3E 0049 R 06                       CMP     CRT_MODE,6
F230  75 06                                 JNE     C77             ; HANDLE IF LOW RES
F232  BB 0180                       C76:    MOV     BX,180H
F235  B9 0703                               MOV     CX,703H         ; SET PARMS FOR HIGH RES
                                    ;------ DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
F238  22 EA                         C77:    AND     CH,DL           ; ADDRESS OF PEL WITHIN BYTE TO CH
                                    ;------ DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
F23A  D3 EA                                 SHR     DX,CL           ; SHIFT BY CORRECT AMOUNT
F23C  03 F2                                 ADD     SI,DX           ; INCREMENT THE POINTER
F23E  80 3E 0049 R 0A                       CMP     CRT_MODE,0AH    ; 640X200 4 COLOR?
F243  75 02                                 JNE     C78             ; NO, JUMP
F245  03 F2                                 ADD     SI,DX           ; INCREMENT THE POINTER
F247  8A F7                         C78:    MOV     DH,BH           ; GET THE # OF BITS IN RESULT TO DH
                                    ;------ MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
F249  2A C9                                 SUB     CL,CL           ; ZERO INTO STORAGE LOCATION
F24B  D0 C8                         C79:    ROR     AL,1            ; LEFT JUSTIFY THE VALUE IN AL
                                                                    ; (FOR WRITE)
F24D  02 CD                                 ADD     CL,CH           ; ADD IN THE BIT OFFSET VALUE
F24F  FE CF                                 DEC     BH              ; LOOP CONTROL
F251  75 F8                                 JNZ     C79             ; ON EXIT, CL HAS SHIFT COUNT TO
                                                                    ; RESTORE BITS
F253  8A E3                                 MOV     AH,BL           ;  GET MASK TO AH
F255  D2 EC                                 SHR     AH,CL           ;  MOVE THE MASK TO CORRECT
                                                                    ; LOCATION
F257  5B                                    POP     BX              ;  RECOVER REG
F258  C3                                    RET                     ;  RETURN WITH EVERYTHING SET UP
F259                                C72      ENDP
                                    ;--------------------------------------------------------
                                    ;  SCROLL UP
                                    ;   THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
                                    ; ENTRY --
                                    ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                                    ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                                    ;   BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                                    ;  BH = FILL VALUE FOR BLANKED LINES
                                    ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                                    ;  DS = DATA SEGMENT
                                    ;  ES = REGEN SEGMENT
                                    ; EXIT --
                                    ;  NOTHING, THE SCREEN IS SCROLLED
                                    ;----------------------------------------------------------
F259                                GRAPHICS_UP     PROC    NEAR
F259  8A D8                                 MOV     BL,AL   ; SAVE LINE COUNT IN BL
F25B  8B C1                                 MOV     AX,CX   ; GET UPPER LEFT POSITION INTO AX REG
                                    ;------ USE CHARACTER SUBROUTINE FOR POSITIONING
                                    ;------ ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
F25D  E8 F72C R                             CALL    GRAPH_POSN
F260  8B F8                                 MOV     DI,AX           ; SAVE RESULT AS DESTINATION
                                                                    ; ADDRESS
                                    ;------ DETERMINE SIZE OF WINDOW
F262  2B D1                                 SUB     DX,CX
F264  81 C2 0101                            ADD     DX,101H         ; ADJUST VALUES
F268  D0 E6                                 SAL     DH,1            ; MULTIPLY # ROWS BY 4 SINCE 8 VERT
                                                                    ; DOTS/CHAR
F26A  D0 E6                                 SAL     DH,1            ;  AND EVEN/ODD ROWS
                                    ;------ DETERMINE CRT MODE
F26C  80 3E 0049 R 06                       CMP     CRT_MODE,6      ; TEST FOR HIGH RES
F271  74 1D                                 JE      C80             ; FIND_SOURCE
                                    ;------ MEDIUM RES UP
F273  D0 E2                                 SAL     DL,1            ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
F275  D1 E7                                 SAL     DI,1            ; OFFSET *2 SINCE 2 BYTES/CHAR
F277  80 3E 0049 R 04                       CMP     CRT_MODE,4      ; TEST FOR MEDIUM RES
F27C  74 12                                 JE      C80
F27E  80 3E 0049 R 05                       CMP     CRT_MODE,5      ; TEST FOR MEDIUM RES
F283  74 0B                                 JE      C80
F285  80 3E 0049 R 0A                       CMP     CRT_MODE,0AH    ; TEST FOR MEDIUM RES
F28A  74 04                                 JE      C80
                                    ;------ LOW RES UP
F28C  D0 E2                                 SAL     DL,1            ; # COLUMNS * 2 AGAIN, SINCE 4
                                                                    ; BYTES/CHAR
F28E  D1 E7                                 SAL     DI,1            ; OFFSET *2 AGAIN, SINCE 4
                                                                    ; BYTES/CHAR
```

```
                                ;------ DETERMINE THE SOURCE ADDRESS IN THE BUFFER
F290                            C80:                              ; FIND_SOURCE
F290  06                                PUSH    ES                ; GET SEGMENTS BOTH POINTING TO
                                                                  ; REGEN
F291  1F                                POP     DS
F292  2A ED                             SUB     CH,CH             ; ZERO TO HIGH OF COUNT REG
F294  D0 E3                             SAL     BL,1              ; MULTIPLY NUMBER OF LINES BY 4
F296  D0 E3                             SAL     BL,1
F298  74 67                             JZ      C86               ; IF ZERO, THEN BLANK ENTIRE FIELD
F29A  8A C3                             MOV     AL,BL             ; GET NUMBER OF LINES IN AL
F29C  B4 50                             MOV     AH,80             ; 80 BYTES/ROW
F29E  F6 E4                             MUL     AH                ; DETERMINE OFFSET TO SOURCE
F2A0  8B F7                             MOV     SI,DI             ; SET UP SOURCE
F2A2  03 F0                             ADD     SI,AX             ;  ADD IN OFFSET TO IT
F2A4  8A E6                             MOV     AH,DH             ; NUMBER OF ROWS IN FIELD
F2A6  2A E3                             SUB     AH,BL             ; DETERMINE NUMBER TO MOVE
                                ;------ LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD
                                                                  ; FIELDS
F2A8                            C81:                              ; ROW_LOOP
F2A8  E8 F3C7 R                         CALL    C95               ; MOVE ONE ROW
F2AB  1E                                PUSH    DS                ; SAVE DATA SEG
F2AC  E8 138B R                         CALL    DDS               ; POINT TO BIOS DATA AREA
F2AF  80 3E 0049 R 09                   CMP     CRT_MODE,9        ; MODE USES 32K REGEN?
F2B4  1F                                POP     DS                ; RESTORE DATA SEG
F2B5  72 15                             JC      C82               ; NO, JUMP
F2B7  81 C6 2000                        ADD     SI,2000H          ; ADJUST POINTERS
F2BB  81 C7 2000                        ADD     DI,2000H
F2BF  E8 F3C7 R                         CALL    C95               ; MOVE 2 MORE ROWS
F2C2  81 EE 3FB0                        SUB     SI,4000H-80       ; BACK UP POINTERS
F2C6  81 EF 3FB0                        SUB     DI,4000H-80       ;
F2CA  FE CC                             DEC     AH                ; ADJUST COUNT
F2CC  81 EE 1FB0                C82:    SUB     SI,2000H-80       ; MOVE TO NEXT ROW
F2D0  81 EF 1FB0                        SUB     DI,2000H-80
F2D4  FE CC                             DEC     AH                ; NUMBER OF ROWS TO MOVE
F2D6  75 D0                             JNZ     C81               ; CONTINUE TILL ALL MOVED
                                ;------ FILL IN THE VACATED LINE(S)
F2D8                            C83:                              ; CLEAR_ENTRY
F2D8  8A C7                             MOV     AL,BH             ; ATTRIBUTE TO FILL WITH
F2DA  E8 F3E0 R                 C84:    CALL    C96               ; CLEAR THAT ROW
F2DD  1E                                PUSH    DS                ; SAVE DATA SEG
F2DE  E8 138B R                         CALL    DDS               ; POINT TO BIOS DATA AREA
F2E1  80 3E 0049 R 09                   CMP     CRT_MODE,9        ; MODE USES 32K REGEN?
F2E6  1F                                POP     DS                ; RESTORE DATA SEG
F2E7  72 0D                             JC      C85               ; NO, JUMP
F2E9  81 C7 2000                        ADD     DI,2000H
F2ED  E8 F3E0 R                         CALL    C96               ; CLEAR 2 MORE ROWS
F2F0  81 EF 3FB0                        SUB     DI,4000H-80       ; BACK UP POINTERS
F2F4  FE CB                             DEC     BL                ; ADJUST COUNT
F2F6  81 EF 1FB0                C85:    SUB     DI,2000H-80       ; POINT TO NEXT LINE
F2FA  FE CB                             DEC     BL                ; NUMBER OF LINES TO FILL
F2FC  75 DC                             JNZ     C84               ; CLEAR_LOOP
F2FE  E9 0F70 R                         JMP     VIDEO_RETURN      ; EVERYTHING DONE
F301                            C86:                              ; BLANK_FIELD
F301  8A DE                             MOV     BL,DH             ; SET BLANK COUNT TO EVERYTHING IN
                                                                  ; FIELD
F303  EB D3                             JMP     C83               ; CLEAR THE FIELD
F305                            GRAPHICS_UP     ENDP
                                ;----------------------------------------------------
                                ; SCROLL DOWN
                                ;  THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
                                ; ENTRY --
                                ;  CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
                                ;  DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
                                ;   BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
                                ;  BH = FILL VALUE FOR BLANKED LINES
                                ;  AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
                                ;  DS = DATA SEGMENT
                                ;  ES = REGEN SEGMENT
                                ; EXIT --
                                ;  NOTHING, THE SCREEN IS SCROLLED
                                ;-----------------------------------------------------
F305                            GRAPHICS_DOWN   PROC    NEAR
F305  FD                                STD               ; SET DIRECTION
F306  8A D8                             MOV     BL,AL     ; SAVE LINE COUNT IN BL
F308  8B C2                             MOV     AX,DX     ; GET LOWER RIGHT POSITION INTO AX REG
                                ;------ USE CHARACTER SUBROUTINE FOR POSITIONING
                                ;------ ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
F30A  E8 F72C R                         CALL    GRAPH_POSN
F30D  8B F8                             MOV     DI,AX             ; SAVE RESULT AS DESTINATION
                                                                  ; ADDRESS
                                ;------ DETERMINE SIZE OF WINDOW
F30F  2B D1                             SUB     DX,CX
F311  81 C2 0101                        ADD     DX,101H           ; ADJUST VALUES
F315  D0 E6                             SAL     DH,1              ; MULTIPLY # ROWS BY 4 SINCE 8 VERT
                                                                  ; DOTS/CHAR
F317  D0 E6                             SAL     DH,1              ;  AND EVEN/ODD ROWS
                                ;------ DETERMINE CRT MODE
F319  80 3E 0049 R 06                   CMP     CRT_MODE,6        ; TEST FOR HIGH RES
F31E  74 22                             JZ      C87               ; FIND_SOURCE_DOWN
```

```
                                 ;------ MEDIUM RES DOWN
F320  D0 E2                              SAL     DL,1             ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
                                                                  ; (OFFSET OK)
F322  D1 E7                              SAL     DI,1             ; OFFSET *2 SINCE 2 BYTES/CHAR
F324  47                                 INC     DI               ; POINT TO LAST BYTE
F325  80 3E 0049 R 04                    CMP     CRT_MODE,4       ; TEST FOR MEDIUM RES
F32A  74 16                              JZ      C87              ; FIND_SOURCE_DOWN
F32C  80 3E 0049 R 05                    CMP     CRT_MODE,5       ; TEST FOR MEDIUM RES
F331  74 0F                              JZ      C87              ; FIND_SOURCE_DOWN
F333  80 3E 0049 R 0A                    CMP     CRT_MODE,0AH     ; TEST FOR MEDIUM RES
F338  74 08                              JZ      C87              ; FIND_SOURCE_DOWN
F33A  4F                                 DEC     DI
F33B  D0 E2                              SAL     DL,1             ; # COLUMNS * 2 AGAIN, SINCE 4
                                                                  ; BYTES/CHAR (OFFSET OK)
F33D  D1 E7                              SAL     DI,1             ; OFFSET *2 AGAIN, SINCE 4
                                                                  ; BYTES/CHAR
F33F  83 C7 03                           ADD     DI,3             ; POINT TO LAST BYTE
                                 ;------ DETERMINE THE SOURCE ADDRESS IN THE BUFFER
F342                             C87:                             ; FIND_SOURCE_DOWN
F342  2A ED                              SUB     CH,CH            ; ZERO TO HIGH OF COUNT REG
F344  B8 00F0                            MOV     AX,240           ; OFFSET TO LAST ROW OF PIXELS IF
                                                                  ; 16K REGEN
F347  80 3E 0049 R 09                    CMP     CRT_MODE,9       ; USING 32K REGEN?
F34C  72 03                              JC      C88              ; NO, JUMP
F34E  B8 00A0                            MOV     AX,160           ; OFFSET TO LAST ROW OF PIXELS IF
                                                                  ; 32K REGEN
F351  03 F8                      C88:    ADD     DI,AX            ; POINT TO LAST ROW OF PIXELS
F353  D0 E3                              SAL     BL,1             ; MULTIPLY NUMBER OF LINES BY 4
F355  D0 E3                              SAL     BL,1
F357  74 6A                              JZ      C94              ; IF ZERO, THEN BLANK ENTIRE FIELD
F359  8A C3                              MOV     AL,BL            ; GET NUMBER OF LINES IN AL
F35B  B4 50                              MOV     AH,80            ; 80 BYTES/ROW
F35D  F6 E4                              MUL     AH               ; DETERMINE OFFSET TO SOURCE
F35F  8B F7                              MOV     SI,DI            ; SET UP SOURCE
F361  2B F0                              SUB     SI,AX            ;  SUBTRACT THE OFFSET
F363  8A E6                              MOV     AH,DH            ; NUMBER OF ROWS IN FIELD
F365  2A E3                              SUB     AH,BL            ; DETERMINE NUMBER TO MOVE
F367  06                                 PUSH    ES               ; BOTH SEGMENTS TO REGEN
F368  1F                                 POP     DS
                                 ;------ LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD
                                 ;       FIELDS
F369                             C89:                             ; ROW_LOOP_DOWN
F369  E8 F3C7 R                          CALL    C95              ; MOVE ONE ROW
F36C  1E                                 PUSH    DS               ; SAVE DATA SEG
F36D  E8 138B R                          CALL    DDS              ; POINT TO BIOS DATA AREA
F370  80 3E 0049 R 09                    CMP     CRT_MODE,9       ; MODE USES 32K REGEN?
F375  1F                                 POP     DS               ; RESTORE DATA SEG
F376  72 15                              JC      C90              ; NO, JUMP
F378  81 C6 2000                         ADD     SI,2000H         ; ADJUST POINTERS
F37C  81 C7 2000                         ADD     DI,2000H
F380  E8 F3C7 R                          CALL    C95              ; MOVE 2 MORE ROWS
F383  81 EE 4050                         SUB     SI,4000H+80      ; BACK UP POINTERS
F387  81 EF 4050                         SUB     DI,4000H+80      ;
F38B  FE CC                              DEC     AH               ; ADJUST COUNT
F38D  81 EE 2050                 C90:    SUB     SI,2000H+80      ; MOVE TO NEXT ROW
F391  81 EF 2050                         SUB     DI,2000H+80
F395  FE CC                              DEC     AH               ; NUMBER OF ROWS TO MOVE
F397  75 D0                              JNZ     C89              ; CONTINUE TILL ALL MOVED
                                 ;------ FILL IN THE VACATED LINE(S)
F399                             C91:                             ; CLEAR_ENTRY_DOWN
F399  8A C7                              MOV     AL,BH            ; ATTRIBUTE TO FILL WITH
F39B                             C92:                             ; CLEAR_LOOP_DOWN
F39B  E8 F3E0 R                          CALL    C96              ; CLEAR A ROW
F39E  1E                                 PUSH    DS               ; SAVE DATA SEG
F39F  E8 138B R                          CALL    DDS              ; POINT TO BIOS DATA AREA
F3A2  80 3E 0049 R 09                    CMP     CRT_MODE,9       ; MODE USES 32K REGEN?
F3A7  1F                                 POP     DS               ; RESTORE DATA SEG
F3A8  72 0D                              JC      C93              ; NO, JUMP
F3AA  81 C7 2000                         ADD     DI,2000H
F3AE  E8 F3E0 R                          CALL    C96              ; CLEAR 2 MORE ROWS
F3B1  81 EF 4050                         SUB     DI,4000H+80      ; BACK UP POINTERS
F3B5  FE CB                              DEC     BL               ; ADJUST COUNT
F3B7  81 EF 2050                 C93:    SUB     DI,2000H+80      ; POINT TO NEXT LINE
F3BB  FE CB                              DEC     BL               ; NUMBER OF LINES TO FILL
F3BD  75 DC                              JNZ     C92              ; CLEAR_LOOP_DOWN
F3BF  FC                                 CLD                      ; RESET THE DIRECTION FLAG
F3C0  E9 0F70 R                          JMP     VIDEO_RETURN     ; EVERYTHING DONE
F3C3                             C94:                             ; BLANK_FIELD_DOWN
F3C3  8A DE                              MOV     BL,DH            ; SET BLANK COUNT TO EVERYTHING IN
                                                                  ; FIELD
F3C5  EB D2                              JMP     C91              ; CLEAR THE FIELD
F3C7                             GRAPHICS_DOWN   ENDP
                                 ;------ ROUTINE TO MOVE ONE ROW OF INFORMATION
F3C7                             C95     PROC    NEAR
F3C7  8A CA                              MOV     CL,DL            ; NUMBER OF BYTES IN THE ROW
F3C9  56                                 PUSH    SI
F3CA  57                                 PUSH    DI               ; SAVE POINTERS
F3CB  F3/ A4                             REP     MOVSB            ; MOVE THE EVEN FIELD
F3CD  5F                                 POP     DI
F3CE  5E                                 POP     SI
F3CF  81 C6 2000                         ADD     SI,2000H
F3D3  81 C7 2000                         ADD     DI,2000H         ; POINT TO THE ODD FIELD
F3D7  56                                 PUSH    SI
F3D8  57                                 PUSH    DI               ; SAVE THE POINTERS
F3D9  8A CA                              MOV     CL,DL            ; COUNT BACK
F3DB  F3/ A4                             REP     MOVSB            ; MOVE THE ODD FIELD
F3DD  5F                                 POP     DI
F3DE  5E                                 POP     SI               ; POINTERS BACK
F3DF  C3                                 RET                      ; RETURN TO CALLER
F3E0                             C95     ENDP
```

```
                                ;------ CLEAR A SINGLE ROW
F3E0                            C96     PROC    NEAR
F3E0  8A CA                             MOV     CL,DL           ; NUMBER OF BYTES IN FIELD
F3E2  57                                PUSH    DI              ; SAVE POINTER
F3E3  F3/ AA                            REP     STOSB           ; STORE THE NEW VALUE
F3E5  5F                                POP     DI              ; POINTER BACK
F3E6  81 C7 2000                        ADD     DI,2000H        ; POINT TO ODD FIELD
F3EA  57                                PUSH    DI
F3EB  8A CA                             MOV     CL,DL
F3ED  F3/ AA                            REP     STOSB           ; FILL THE ODD FILELD
F3EF  5F                                POP     DI
F3F0  C3                                RET                     ; RETURN TO CALLER
F3F1                            C96     ENDP
                                ;---------------------------------------------------
                                ; GRAPHICS WRITE
                                ;  THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT
                                ;  POSITION ON THE SCREEN.
                                ; ENTRY --
                                ;  AL = CHARACTER TO WRITE
                                ;  BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
                                ;       IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER
                                ;       (0 IS USED FOR THE BACKGROUND COLOR)
                                ;  CX = NUMBER OF CHARS TO WRITE
                                ;  DS = DATA SEGMENT
                                ;  ES = REGEN SEGMENT
                                ; EXIT --
                                ;  NOTHING IS RETURNED
                                ;
                                ; GRAPHICS READ
                                ;   THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR
                                ;   POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO
                                ;   THE CHARACTER GENERATOR CODE POINTS
                                ; ENTRY --
                                ;  NONE  (0 IS ASSUMED AS THE BACKGROUND COLOR)
                                ; EXIT --
                                ;  AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
                                ;
                                ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN
                                ; ROM.  INTERRUPT 44H IS USED TO POINT TO THE TABLE FOR THE FIRST
                                ; 128 CHARS.  INTERRUPT 1FH IS USED TO POINT TO THE TABLE FOR THE
                                ; SECOND 128 CHARS.
                                ;------------------------------------------------------
                                        ASSUME  CS:CODE,DS:DATA,ES:DATA
F3F1                            GRAPHICS_WRITE  PROC    NEAR
F3F1  32 E4                             XOR     AH,AH           ; ZERO TO HIGH OF CODE POINT
F3F3  50                                PUSH    AX              ; SAVE CODE POINT VALUE
                                ;------ DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
F3F4  E8 F729 R                         CALL    R59             ; FIND LOCATION IN REGEN BUFFER
F3F7  8B F8                             MOV     DI,AX           ; REGEN POINTER IN DI
                                ;------ DETERMINE REGION TO GET CODE POINTS FROM
F3F9  58                                POP     AX              ; RECOVER CODE POINT
F3FA  BE 0110 R                         MOV     SI,OFFSET CSET_PTR ; ASSUME FIRST HALF
F3FD  3C 80                             CMP     AL,80H          ; IS IT IN FIRST HALF?
F3FF  72 05                             JB      R1              ; JUMP IF IT IS
F401  BE 007C R                         MOV     SI,OFFSET EXT_PTR ; SET POINTER FOR SECOND HALF
F404  2C 80                             SUB     AL,80H          ; ZERO ORIGIN FOR SECOND HALF
F406                            R1:                             ; EXTEND_CHAR
F406  1E                                PUSH    DS              ; SAVE DATA POINTER
F407  33 D2                             XOR     DX,DX
F409  8E DA                             MOV     DS,DX           ; ESTABLISH VECTOR ADDRESSING
                                        ASSUME  DS:ABS0
F40B  C5 34                             LDS     SI,DWORD PTR [SI] ; GET THE OFFSET OF THE TABLE
F40D  8C DA                             MOV     DX,DS           ; GET THE SEGMENT OF THE TABLE
                                        ASSUME  DS:DATA
F40F  1F                                POP     DS              ; RECOVER DATA SEGMENT
F410  52                                PUSH    DX              ; SAVE TABLE SEGMENT ON STACK
                                ;------ DETERMINE GRAPHICS MODE IN OPERATION
F411  D1 E0                             SAL     AX,1            ; MULTIPLY CODE POINT
F413  D1 E0                             SAL     AX,1            ;  VALUE BY 8
F415  D1 E0                             SAL     AX,1
F417  03 F0                             ADD     SI,AX           ; SI HAS OFFSET OF DESIRED CODES
F419  80 3E 0049 R 04                   CMP     CRT_MODE,4
F41E  74 45                             JE      R9              ; TEST FOR MEDIUM RESOLUTION MODE
F420  80 3E 0049 R 05                   CMP     CRT_MODE,5
F425  74 3E                             JE      R9              ; TEST FOR MEDIUM RESOLUTION MODE
F427  80 3E 0049 R 0A                   CMP     CRT_MODE,0AH
F42C  75 03                             JNE     R3              ; TEST FOR MEDIUM RESOLUTION MODE
F42E  E9 F4D4 R                         JMP     R16
F431  80 3E 0049 R 06           R3:     CMP     CRT_MODE,6      ; TEST FOR HIGH RESOLUTION MODE
F436  75 53                             JNE     R12             ; GOTO LOW RESOLUTION IF NOT
                                ;------ HIGH RESOLUTION MODE
F438  1F                                POP     DS              ; RECOVER TABLE POINTER SEGMENT
F439  57                        R5:     PUSH    DI              ; SAVE REGEN POINTER
F43A  56                                PUSH    SI              ; SAVE CODE POINTER
F43B  B6 04                             MOV     DH,4            ; NUMBER OF TIMES THROUGH LOOP
F43D  AC                        R6:     LODSB                   ; GET BYTE FROM CODE POINTS
F43E  F6 C3 80                          TEST    BL,80H          ; SHOULD WE USE THE FUNCTION
F441  75 16                             JNZ     R8              ;  TO PUT CHAR IN?
F443  AA                                STOSB                   ; STORE IN REGEN BUFFER
F444  AC                                LODSB
F445  26: 88 85 1FFF            R7:     MOV     ES:[DI+2000H-1],AL ; STORE IN SECOND HALF
F44A  83 C7 4F                          ADD     DI,79           ; MOVE TO NEXT ROW IN REGEN
F44D  FE CE                             DEC     DH              ; DONE WITH LOOP
F44F  75 EC                             JNZ     R6
F451  5E                                POP     SI
F452  5F                                POP     DI              ; RECOVER REGEN POINTER
F453  47                                INC     DI              ; POINT TO NEXT CHAR POSITION
F454  E2 E3                             LOOP    R5              ; MORE CHARS TO WRITE
```

```
F456  E9 0F70 R               R705:   JMP     VIDEO_RETURN
F459  26: 32 05               R8:     XOR     AL,ES:[DI]      ; EXCLUSIVE OR WITH CURRENT DATA
F45C  AA                              STOSB                   ; STORE THE CODE POINT
F45D  AC                              LODSB                   ; AGAIN FOR ODD FIELD
F45E  26: 32 85 1FFF                  XOR     AL,ES:[DI+2000H-1]
F463  EB E0                           JMP     R7              ; BACK TO MAINSTREAM
                              ;------ MEDIUM RESOLUTION WRITE
F465                          R9:                             ; MED_RES_WRITE
F465  1F                              POP     DS              ; RECOVER TABLE POINTER SEGMENT
F466  8A D3                           MOV     DL,BL           ; SAVE HIGH COLOR BIT
F468  D1 E7                           SAL     DI,1            ; OFFSET*2 SINCE 2 BYTES/CHAR
F46A  E8 F659 R                       CALL    R40             ; EXPAND BL TO FULL WORD OF COLOR
F46D                          R10:                            ; MED_CHAR
F46D  57                              PUSH    DI              ; SAVE REGEN POINTER
F46E  56                              PUSH    SI              ; SAVE THE CODE POINTER
F46F  B6 04                           MOV     DH,4            ; NUMBER OF LOOPS
F471  E8 F626 R               R11:    CALL    R35             ; DO FIRST 2 BYTES
F474  81 C7 2000                      ADD     DI,2000H        ; NEXT SPOT IN REGEN
F478  E8 F626 R                       CALL    R35             ; DO NEXT 2 BYTES
F47B  81 EF 1FB0                      SUB     DI,2000H-80
F47F  FE CE                           DEC     DH
F481  75 EE                           JNZ     R11             ; KEEP GOING
F483  5E                              POP     SI              ; RECOVER CODE PONTER
F484  5F                              POP     DI              ; RECOVER REGEN POINTER
F485  47                              INC     DI              ; POINT TO NEXT CHAR POSITION
F486  47                              INC     DI
F487  E2 E4                           LOOP    R10             ; MORE TO WRITE
F489  EB CB                           JMP     R705
                              ;------ LOW RESOLUTION WRITE
F48B                          R12:                            ; LOW_RES_WRITE
F48B  1F                              POP     DS              ; RECOVER TABLE POINTER SEGMENT
F48C  8A D3                           MOV     DL,BL           ; SAVE HIGH COLOR BIT
F48E  D1 E7                           SAL     DI,1            ; OFFSET*4 SINCE 4 BYTES/CHAR
F490  D1 E7                           SAL     DI,1            ;
F492  E8 F66E R                       CALL    R42             ; EXPAND BL TO FULL WORD OF COLOR
F495                          R13:                            ; MED_CHAR
F495  57                              PUSH    DI              ; SAVE REGEN POINTER
F496  56                              PUSH    SI              ; SAVE THE CODE POINTER
F497  B6 04                           MOV     DH,4            ; NUMBER OF LOOPS
F499  E8 F645 R               R14:    CALL    R39             ; EXPAND DOT ROW IN REGEN
F49C  81 C7 2000                      ADD     DI,2000H        ; POINT TO NEXT REGEN ROW
F4A0  E8 F645 R                       CALL    R39             ; EXPAND DOT ROW IN REGEN
F4A3  1E                              PUSH    DS              ; SAVE DS
F4A4  E8 138B R                       CALL    DDS             ; POINT TO BIOS DATA AREA
F4A7  80 3E 0049 R 09                 CMP     CRT_MODE,09H    ; USING 32K REGEN AREA?
F4AC  1F                              POP     DS              ; RECOVER DS
F4AD  75 14                           JNE     R15             ; JUMP IF 16K REGEN
F4AF  81 C7 2000                      ADD     DI,2000H        ; POINT TO NEXT REGEN ROW
F4B3  E8 F645 R                       CALL    R39             ; EXPAND DOT ROW IN REGEN
F4B6  81 C7 2000                      ADD     DI,2000H        ; POINT TO NEXT REGEN ROW
F4BA  E8 F645 R                       CALL    R39             ; EXPAND DOT ROW IN REGEN
F4BD  81 EF 3FB0                      SUB     DI,4000H-80     ; ADJUST REGEN POINTER
F4C1  FE CE                           DEC     DH
F4C3  81 EF 1FB0              R15:    SUB     DI,2000H-80     ; ADJUST REGEN POINTER TO NEXT ROW
F4C7  FE CE                           DEC     DH
F4C9  75 CE                           JNZ     R14             ; KEEP GOING
F4CB  5E                              POP     SI              ; RECOVER CODE PONTER
F4CC  5F                              POP     DI              ; RECOVER REGEN POINTER
F4CD  83 C7 04                        ADD     DI,4            ; POINT TO NEXT CHAR POSITION
F4D0  E2 C3                           LOOP    R13             ; MORE TO WRITE
F4D2  EB 82                           JMP     R705
                              ;----- 640X200 4 COLOR GRAPHICS WRITE
F4D4  1F                      R16:    POP     DS              ; RECOVER TABLE SEGMENT POINTER
F4D5  8A D3                           MOV     DL,BL           ; SAVE HIGH COLOR BIT
F4D7  D1 E7                           SAL     DI,1            ; OFFSET*2 SINCE 2 BYTES/CHAR
                              ; EXPAND LOW 2 COLOR BITS IN BL (c1c0)
                              ; INTO BX (c0c0c0c0c0c0c0c0c1c1c1c1c1c1c1c1)
F4D9  33 C0                           XOR     AX,AX
F4DB  F6 C3 01                        TEST    BL,1            ; c0 COLOR BIT ON?
F4DE  74 02                           JZ      R17             ; NO, JUMP
F4E0  B4 FF                           MOV     AH,0FFH         ; YES, SET ALL c0 BITS ON
F4E2  F6 C3 02                R17:    TEST    BL,2            ; c1 COLOR BIT ON?
F4E5  74 02                           JZ      R18             ; NO, JUMP
F4E7  B0 FF                           MOV     AL,0FFH         ; YES, SET ALL c1 BITS ON
F4E9  8B D8                   R18:    MOV     BX,AX           ; COLOR MASK IN BX
F4EB                          R19:
F4EB  57                              PUSH    DI              ; SAVE REGEN POINTER
F4EC  56                              PUSH    SI              ; SAVE CODE POINT POINTER
F4ED  B6 02                           MOV     DH,2            ; SET LOOP COUNTER
F4EF  E8 F518 R               R20:    CALL    R21             ; DO FIRST DOT ROW
F4F2  81 C7 2000                      ADD     DI,2000H        ; ADJUST REGEN POINTER
F4F6  E8 F518 R                       CALL    R21             ; DO NEXT DOT ROW
F4F9  81 C7 2000                      ADD     DI,2000H        ; ADJUST REGEN POINTER
F4FD  E8 F518 R                       CALL    R21             ; DO NEXT DOT ROW
F500  81 C7 2000                      ADD     DI,2000H        ; ADJUST REGEN POINTER
F504  E8 F518 R                       CALL    R21             ; DO NEXT DOT ROW
F507  81 EF 5F60                      SUB     DI,6000H-160    ; ADJUST REGEN POINTER TO NEXT ROW
F50B  FE CE                           DEC     DH
F50D  75 E0                           JNZ     R20             ; KEEP GOING
F50F  5E                              POP     SI              ; RECOVER CODE POINT POINTER
F510  5F                              POP     DI              ; RECOVER REGEN POINTER
F511  47                              INC     DI              ; POINT TO NEXT CHARACTER
F512  47                              INC     DI
F513  E2 D6                           LOOP    R19             ; MORE TO WRITE
F515  E9 0F70 R                       JMP     VIDEO_RETURN
```

```
F518                          R21     PROC    NEAR
F518  AC                              LODSB                   ; GET CODE POINT
F519  8A E0                           MOV     AH,AL           ; COPY INTO AH
F51B  23 C3                           AND     AX,BX           ; SET COLOR
F51D  F6 C2 80                        TEST    DL,80H          ; XOR FUNCTION?
F520  74 07                           JZ      R22             ; NO, JUMP
F522  26: 32 25                       XOR     AH,ES:[DI]      ; EXCLUSIVE OR WITH CURRENT DATA
F525  26: 32 45 01                    XOR     AL,ES:[DI+1]
F529  26: 88 25               R22:    MOV     ES:[DI],AH      ; STORE IN REGEN BUFFER
F52C  26: 88 45 01                    MOV     ES:[DI+1],AL
F530  C3                              RET
F531                          R21   ENDP
F531                          GRAPHICS_WRITE  ENDP
                              ;---------------------------------------
                              ; GRAPHICS READ
                              ;---------------------------------------
F531                          GRAPHICS_READ   PROC     NEAR
F531  E8 F729 R                       CALL    R59             ; CONVERTED TO OFFSET IN REGEN
F534  8B F0                           MOV     SI,AX           ; SAVE IN SI
F536  83 EC 08                        SUB     SP,8            ; ALLOCATE SPACE TO SAVE THE READ
                                                              ; CODE POINT
F539  8B EC                           MOV     BP,SP           ; POINTER TO SAVE AREA
                              ;------ DETERMINE GRAPHICS MODES
F53B  06                              PUSH    ES
F53C  B6 04                           MOV     DH,4            ; NUMBER OF PASSES
F53E  80 3E 0049 R 06                 CMP     CRT_MODE,6
F543  74 17                           JZ      R23             ; HIGH RESOLUTION
F545  80 3E 0049 R 04                 CMP     CRT_MODE,4
F54A  74 61                           JZ      R28             ; MEDIUM RESOLUTION
F54C  80 3E 0049 R 05                 CMP     CRT_MODE,5
F551  74 5A                           JZ      R28             ; MEDIUM RESOLUTION
F553  80 3E 0049 R 0A                 CMP     CRT_MODE,0AH
F558  74 53                           JZ      R28             ; MEDIUM RESOLUTION
F55A  EB 18                           JMP     SHORT R25       ; LOW RESOLUTION
                              ;------ HIGH RESOLUTION READ
                              ;------ GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
F55C  1F                      R23:    POP     DS              ; POINT TO REGEN SEGMENT
F55D  8A 04                   R24:    MOV     AL,[SI]         ; GET FIRST BYTE
F55F  88 46 00                        MOV     [BP],AL         ; SAVE IN STORAGE AREA
F562  45                              INC     BP              ; NEXT LOCATION
F563  8A 84 2000                      MOV     AL,[SI+2000H]   ; GET LOWER REGION BYTE
F567  88 46 00                        MOV     [BP],AL         ; ADJUST AND STORE
F56A  45                              INC     BP
F56B  83 C6 50                        ADD     SI,80           ; POINTER INTO REGEN
F56E  FE CE                           DEC     DH              ; LOOP CONTROL
F570  75 EB                           JNZ     R24             ; DO IT SOME MORE
F572  EB 6E                           JMP     SHORT R31       ; GO MATCH THE SAVED CODE POINTS
                              ;------ LOW RESOLUTION READ
F574  1F                      R25:    POP     DS              ; POINT TO REGEN SEGMENT
F575  D1 E6                           SAL     SI,1            ; OFFSET*4 SINCE 4 BYTES/CHAR
F577  D1 E6                           SAL     SI,1
F579  E8 F6FC R               R26:    CALL    R55             ; GET 4 BYTES FROM REGEN INTO
                                                              ; SINGLE SAVE
F57C  81 C6 2000                      ADD     SI,2000H        ; GOTO LOWER REGION
F580  E8 F6FC R                       CALL    R55             ; GET 4 BYTES FROM REGEN INTO
                                                              ; SINGLE SAVE
F583  1E                              PUSH    DS              ; SAVE DS
F584  E8 138B R                       CALL    DDS             ; POINT TO BIOS DATA AREA
F587  80 3E 0049 R 09                 CMP     CRT_MODE,9      ; DO WE HAVE A 32K REGEN AREA?
F58C  1F                              POP     DS
F58D  75 14                           JNE     R27             ; NO, JUMP
F58F  81 C6 2000                      ADD     SI,2000H        ; GOTO LOWER REGION
F593  E8 F6FC R                       CALL    R55             ; GET 4 BYTES FROM REGEN INTO
                                                              ; SINGLE SAVE
F596  81 C6 2000                      ADD     SI,2000H        ; GOTO LOWER REGION
F59A  E8 F6FC R                       CALL    R55             ; GET 4 BYTES FROM REGEN INTO
                                                              ; SINGLE SAVE
F59D  81 EE 3FB0                      SUB     SI,4000H-80     ; ADJUST POINTER
F5A1  FE CE                           DEC     DH
F5A3  81 EE 1FB0              R27:    SUB     SI,2000H-80     ; ADJUST POINTER BACK TO UPPER
F5A7  FE CE                           DEC DH
F5A9  75 CE                           JNZ     R26             ;DO IT SOME MORE
F5AB  EB 35                           JMP     SHORT R31       ; GO MATCH THE SAVED CODE POINTS
                              ;------ MEDIUM RESOLUTION READ
F5AD                          R28:                            ; MED_RES_READ
F5AD  1F                              POP     DS              ; POINT TO REGEN SEGMENT
F5AE  D1 E6                           SAL     SI,1            ; OFFSET*2 SINCE 2 BYTES/CHAR
F5B0  E8 F6C3 R               R29:    CALL    R50             ; GET PAIR BYTES FROM REGEN INTO
                                                              ; SINGLE SAVE
F5B3  81 C6 2000                      ADD     SI,2000H        ; GO TO LOWER REGION
F5B7  E8 F6C3 R                       CALL    R50             ; GET THIS PAIR INTO SAVE
F5BA  1E                              PUSH    DS              ; SAVE DS
F5BB  E8 138B R                       CALL    DDS             ; POINT TO BIOS DATA AREA
F5BE  80 3E 0049 R 0A                 CMP     CRT_MODE,0AH    ; DO WE HAVE A 32K REGEN AREA?
F5C3  1F                              POP     DS
F5C4  75 14                           JNE     R30             ; NO, JUMP
F5C6  81 C6 2000                      ADD     SI,2000H        ; GOTO LOWER REGION
F5CA  E8 F6C3 R                       CALL    R50             ; GET PAIR BYTES FROM REGEN INTO
                                                              ; SINGLE SAVE
F5CD  81 C6 2000                      ADD     SI,2000H        ; GOTO LOWER REGION
F5D1  E8 F6C3 R                       CALL    R50             ; GET PAIR BYTES FROM REGEN INTO
                                                              ; SINGLE SAVE
F5D4  81 EE 3FB0                      SUB     SI,4000H-80     ; ADJUST POINTER
F5D8  FE CE                           DEC     DH
F5DA                          R30:
F5DA  81 EE 1FB0                      SUB     SI,2000H-80     ; ADJUST POINTER BACK INTO UPPER
F5DE  FE CE                           DEC     DH
F5E0  75 CE                           JNZ     R29             ; KEEP GOING UNTIL ALL 8 DONE
```

```
                                 ;-------- SAVE AREA HAS CHARACTER IN IT, MATCH IT
F5E2                             R31:                             ; FIND_CHAR
F5E2  33 C0                             XOR     AX,AX
F5E4  8E D8                             MOV     DS,AX             ; ESTABLISH ADDRESSING TO VECTOR
                                        ASSUME  DS:ABS0
F5E6  C4 3E 0110 R                      LES     DI,CSET_PTR       ; GET POINTER TO FIRST HALF
F5EA  83 ED 08                          SUB     BP,8              ; ADJUST POINTER TO BEGINNING OF
                                                                  ; SAVE AREA
F5ED  8B F5                             MOV     SI,BP
F5EF  FC                                CLD                       ; ENSURE DIRECTION
F5F0  32 C0                             XOR     AL,AL             ; CURRENT CODE POINT BEING MATCHED
F5F2  16                         R32:   PUSH    SS                ; ESTABLISH ADDRESSING TO STACK
F5F3  1F                                POP     DS                ; FOR THE STRING COMPARE
F5F4  BA 0080                           MOV     DX,128            ; NUMBER TO TEST AGAINST
F5F7  56                         R33:   PUSH    SI                ; SAVE AREA POINTER
F5F8  57                                PUSH    DI                ; SAVE CODE POINTER
F5F9  B9 0008                           MOV     CX,8              ; NUMBER OF BYTES TO MATCH
F5FC  F3/ A6                            REPE    CMPSB             ; COMPARE THE 8 BYTES
F5FE  5F                                POP     DI                ; RECOVER THE POINTERS
F5FF  5E                                POP     SI
F600  74 1E                             JZ      R34               ; IF ZERO FLAG SET, THEN MATCH
                                                                  ; OCCURRED
F602  FE C0                             INC     AL                ; NO MATCH, MOVE ON TO NEXT
F604  83 C7 08                          ADD     DI,8              ; NEXT CODE POINT
F607  4A                                DEC     DX                ; LOOP CONTROL
F608  75 ED                             JNZ     R33               ; DO ALL OF THEM
                                 ;------ CHAR NOT MATCHED, MIGHT BE IN SECOND HALF
F60A  0A C0                             OR      AL,AL             ; AL<> 0 IF ONLY 1ST HALF SCANNED
F60C  74 12                             JE      R34               ; IF = 0, THEN ALL HAS BEEN SCANNED
F60E  2B C0                             SUB     AX,AX
F610  8E D8                             MOV     DS,AX             ; ESTABLISH ADDRESSING TO VECTOR
                                        ASSUME  DS:ABS0
F612  C4 3E 007C R                      LES     DI,EXT_PTR        ; GET POINTER
F616  8C C0                             MOV     AX,ES             ; SEE IF THE POINTER REALLY EXISTS
F618  0B C7                             OR      AX,DI             ; IF ALL 0, THEN DOESN'T EXIST
F61A  74 04                             JZ      R34               ; NO SENSE LOOKING
F61C  B0 80                             MOV     AL,128            ; ORIGIN FOR SECOND HALF
F61E  EB D2                             JMP     R32               ; GO BACK AND TRY FOR IT
                                        ASSUME  DS:DATA
                                 ;------ CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
F620  83 C4 08                   R34:   ADD     SP,8              ; READJUST THE STACK, THROW AWAY
                                                                  ; WORK AREA
F623  E9 0F70 R                         JMP     VIDEO_RETURN      ; ALL DONE
F626                             GRAPHICS_READ  ENDP
                                 ;--------
F626                             R35    PROC    NEAR
F626  AC                                LODSB                     ; GET CODE POINT
F627  E8 F67E R                         CALL    R43               ; DOUBLE UP ALL THE BITS
F62A  23 C3                      R36:   AND     AX,BX             ; CONVERT THEM TO FOREGROUND COLOR
                                                                  ; ( 0 BACK )
F62C  F6 C2 80                          TEST    DL,80H            ; IS THIS XOR FUNCTION?
F62F  74 07                             JZ      R37               ; NO, STORE IT IN AS IT IS
F631  26: 32 25                         XOR     AH,ES:[DI]        ; DO FUNCTION WITH HALF
F634  26: 32 45 01                      XOR     AL,ES:[DI+1]      ;  AND WITH OTHER HALF
F638  26: 88 25                  R37:   MOV     ES:[DI],AH        ; STORE FIRST BYTE
F63B  26: 88 45 01                      MOV     ES:[DI+1],AL      ; STORE SECOND BYTE
F63F  C3                                RET
F640                             R35    ENDP
                                 ;--------
F640                             R38    PROC    NEAR
F640  E8 F6A0 R                         CALL    R45               ; QUAD UP THE LOW NIBBLE
F643  EB E5                             JMP     R36
F645                             R38    ENDP
                                 ;-----------
                                 ; EXPAND 1 DOT ROW OF A CHAR INTO 4 BYTES IN THE REGEN BUFFER
                                 ;-------------
F645                             R39    PROC    NEAR
F645  AC                                LODSB                     ; GET CODE POINT
F646  50                                PUSH    AX                ; SAVE
F647  51                                PUSH    CX
F648  B1 04                             MOV     CL,4              ; MOV HIGH NIBBLE TO LOW
F64A  D2 E8                             SHR     AL,CL
F64C  59                                POP     CX
F64D  E8 F640 R                         CALL    R38               ; EXPAND TO 2 BYTES & PUT IN REGEN
F650  58                                POP     AX                ; RECOVER CODE POINT
F651  47                                INC     DI                ; ADJUST REGEN POINTER
F652  47                                INC     DI
F653  E8 F640 R                         CALL    R38               ; EXPAND LOW NIBBLE & PUT IN REGEN
F656  4F                                DEC     DI                ; RESTORE REGEN POINTER
F657  4F                                DEC     DI
F658  C3                                RET
F659                             R39    ENDP
                                 ;-----------------------------------------------
                                 ; EXPAND_MED_COLOR
                                 ;  THIS ROUTINE EXPANDS THE LOW 2 BITS IN BL TO
                                 ;  FILL THE ENTIRE BX REGISTER
                                 ; ENTRY --
                                 ;  BL = COLOR TO BE USED ( LOW 2 BITS )
                                 ; EXIT --
                                 ;  BX = COLOR TO BE USED ( 8 REPLICATIONS OF THE 2 COLOR BITS )
                                 ;-----------------------------------------------
```

```
F659                          R40     PROC    NEAR
F659  80 E3 03                        AND     BL,3            ; ISOLATE THE COLOR BITS
F65C  8A C3                           MOV     AL,BL           ; COPY TO AL
F65E  51                              PUSH    CX              ; SAVE REGISTER
F65F  B9 0003                         MOV     CX,3            ; NUMBER OF TIMES TO DO THIS
F662  D0 E0                   R41:    SAL     AL,1
F664  D0 E0                           SAL     AL,1            ; LEFT SHIFT BY 2
F666  0A D8                           OR      BL,AL           ; ANOTHER COLOR VERSION INTO BL
F668  E2 F8                           LOOP    R41             ; FILL ALL OF BL
F66A  8A FB                           MOV     BH,BL           ; FILL UPPER PORTION
F66C  59                              POP     CX              ; REGISTER BACK
F66D  C3                              RET                     ; ALL DONE
F66E                          R40     ENDP
                              ;---------------------------------------------
                              ; EXPAND_LOW_COLOR
                              ;  THIS ROUTINE EXPANDS THE LOW 4 BITS IN BL TO
                              ;  FILL THE ENTIRE BX REGISTER
                              ; ENTRY --
                              ;  BL = COLOR TO BE USED ( LOW 4 BITS )
                              ; EXIT --
                              ;  BX = COLOR TO BE USED ( 4 REPLICATIONS OF THE 4 COLOR BITS )
                              ;---------------------------------------------
F66E                          R42     PROC    NEAR
F66E  51                              PUSH    CX
F66F  80 E3 0F                        AND     BL,0FH          ; ISOLATE THE COLOR BITS
F672  8A FB                           MOV     BH,BL           ; COPY TO BH
F674  B1 04                           MOV     CL,4            ; MOVE TO HIGH NIBBLE
F676  D2 E7                           SHL     BH,CL           ;
F678  0A FB                           OR      BH,BL           ; MAKE BYTE FROM HIGH AND LOW
                                                              ; NIBBLES
F67A  8A DF                           MOV     BL,BH           ;
F67C  59                              POP     CX
F67D  C3                              RET                     ; ALL DONE
F67E                          R42     ENDP
                              ;------------------------------------------
                              ; EXPAND_BYTE
                              ;  THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
                              ;  OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
                              ;  THE RESULT IS LEFT IN AX
                              ;------------------------------------------
F67E                          R43     PROC    NEAR
F67E  52                              PUSH    DX              ; SAVE REGISTERS
F67F  51                              PUSH    CX
F680  53                              PUSH    BX
F681  2B D2                           SUB     DX,DX           ; RESULT REGISTER
F683  B9 0001                         MOV     CX,1            ; MASK REGISTER
F686  8B D8                   R44:    MOV     BX,AX           ; BASE INTO TEMP
F688  23 D9                           AND     BX,CX           ; USE MASK TO EXTRACT A BIT
F68A  0B D3                           OR      DX,BX           ; PUT INTO RESULT REGISTER
F68C  D1 E0                           SHL     AX,1
F68E  D1 E1                           SHL     CX,1            ; SHIFT BASE AND MASK BY 1
F690  8B D8                           MOV     BX,AX           ; BASE TO TEMP
F692  23 D9                           AND     BX,CX           ; EXTRACT THE SAME BIT
F694  0B D3                           OR      DX,BX           ; PUT INTO RESULT
F696  D1 E1                           SHL     CX,1            ; SHIFT ONLY MASK NOW, MOVING TO
                                                              ; NEXT BASE
F698  73 EC                           JNC     R44             ; USE MASK BIT COMING OUT TO
                                                              ; TERMINATE
F69A  8B C2                           MOV     AX,DX           ; RESULT TO PARM REGISTER
F69C  5B                              POP     BX
F69D  59                              POP     CX              ; RECOVER REGISTERS
F69E  5A                              POP     DX
F69F  C3                              RET                     ; ALL DONE
F6A0                          R43     ENDP
                              ;------------------------------------------
                              ; EXPAND_NIBBLE
                              ;  THIS ROUTINE TAKES THE LOW NIBBLE IN AL AND QUADS ALL
                              ;  OF THE BITS, TURNING THE 4 BITS INTO 16 BITS.
                              ;  THE RESULT IS LEFT IN AX
                              ;------------------------------------------
F6A0                          R45     PROC    NEAR
F6A0  52                              PUSH    DX              ; SAVE REGISTERS
F6A1  33 D2                           XOR     DX,DX           ; RESULT REGISTER
F6A3  A8 08                           TEST    AL,8
F6A5  74 03                           JZ      R46
F6A7  80 CE F0                        OR      DH,0F0H
F6AA  A8 04                   R46:    TEST    AL,4
F6AC  74 03                           JZ      R47
F6AE  80 CE 0F                        OR      DH,0FH
F6B1  A8 02                   R47:    TEST    AL,2
F6B3  74 03                           JZ      R48
F6B5  80 CA F0                        OR      DL,0F0H
F6B8  A8 01                   R48:    TEST    AL,1
F6BA  74 03                           JZ      R49
F6BC  80 CA 0F                        OR      DL,0FH
F6BF  8B C2                   R49:    MOV     AX,DX           ; RESULT TO PARM REGISTER
F6C1  5A                              POP     DX              ; RECOVER REGISTERS
F6C2  C3                              RET                     ; ALL DONE
F6C3                          R45     ENDP
```

```
                               ;-----------------------------------------------
                               ; MED_READ_BYTE
                               ; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
                               ;  COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
                               ;  THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
                               ;  POSITION IN THE SAVE AREA
                               ; ENTRY --
                               ;  SI,DS = POINTER TO REGEN AREA OF INTEREST
                               ;  BX = EXPANDED FOREGROUND COLOR
                               ;  BP = POINTER TO SAVE AREA
                               ; EXIT --
                               ;  BP IS INCREMENT AFTER SAVE
                               ;-------------------------------------------------
F6C3                           R50     PROC    NEAR
F6C3  8A 24                            MOV     AH,[SI]           ; GET FIRST BYTE
F6C5  8A 44 01                         MOV     AL,[SI+1]         ; GET SECOND BYTE
F6C8  1E                               PUSH    DS                ; SAVE DS
F6C9  E8 138B R                        CALL    DDS               ; POINT TO BIOS DATA AREA
F6CC  80 3E 0049 R 0A                  CMP     CRT_MODE,0AH      ; IN 640X200 4 COLOR MODE?
F6D1  1F                               POP     DS                ; RESTORE REGEN SEG
F6D2  75 11                            JNE     R52               ; NO, JUMP
                               ; IN 640X200 4 COLOR MODE, ALL THE c0 BITS ARE IN ONE BYTE, AND ALL
                               ; THE c1 BITS ARE IN THE NEXT BYTE. HERE WE CHANGE THEM BACK TO
                               ; NORMAL c1c0 ADJACENT PAIRS.
F6D4  53                               PUSH    BX                ; SAVE REG
F6D5  B9 0008                          MOV     CX,8              ; SET LOOP COUNTER
F6D8  D0 FC                    R51:    SAR     AH,1              ; c0 BIT INTO CARRY
F6DA  D1 DB                            RCR     BX,1              ;  AND INTO BX
F6DC  D0 F8                            SAR     AL,1              ; c1 BIT INTO CARRY
F6DE  D1 DB                            RCR     BX,1              ;  AND INTO BX
F6E0  E2 F6                            LOOP    R51               ; REPEAT
F6E2  8B C3                            MOV     AX,BX             ; RESULT INTO AX
F6E4  5B                               POP     BX                ; RESTORE BX
F6E5  B9 C000                  R52:    MOV     CX,0C000H         ; 2 BIT MASK TO TEST THE ENTRIES
F6E8  32 D2                            XOR     DL,DL             ; RESULT REGISTER
F6EA  85 C1                    R53:    TEST    AX,CX             ; IS THIS SECTION BACKGROUND?
F6EC  74 01                            JZ      R54               ; IF ZERO, IT IS BACKGROUND
F6EE  F9                               STC                       ; WASN'T, SO SET CARRY
F6EF  D0 D2                    R54:    RCL     DL,1              ; MOVE THAT BIT INTO THE RESULT
F6F1  D1 E9                            SHR     CX,1
F6F3  D1 E9                            SHR     CX,1              ; MOVE THE MASK TO THE RIGHT BY 2
                                                                 ; BITS
F6F5  73 F3                            JNC     R53               ; DO IT AGAIN IF MASK DIDN'T FALL
                                                                 ; OUT
F6F7  88 56 00                         MOV     [BP],DL           ; STORE RESULT IN SAVE AREA
F6FA  45                               INC     BP                ; ADJUST POINTER
F6FB  C3                               RET                       ; ALL DONE
F6FC                           R50     ENDP
                               ;-----------------------------------------------
                               ; LOW_READ_BYTE
                               ; THIS ROUTINE WILL TAKE 4 BYTES FROM THE REGEN BUFFER,
                               ;  COMPARE FOR BACKGROUND COLOR, AND PLACE
                               ;  THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
                               ;  POSITION IN THE SAVE AREA
                               ; ENTRY --
                               ;  SI,DS = POINTER TO REGEN AREA OF INTEREST
                               ;  BP = POINTER TO SAVE AREA
                               ; EXIT --
                               ;  BP IS INCREMENT AFTER SAVE
                               ;-------------------------------------------------
F6FC                           R55     PROC    NEAR
F6FC  8A 24                            MOV     AH,[SI]           ;GET FIRST 2 BYTES
F6FE  8A 44 01                         MOV     AL,[SI+1]
F701  32 D2                            XOR     DL,DL
F703  E8 F714 R                        CALL    R56               ;BUILD HIGH NIBBLE
F706  8A 64 02                         MOV     AH,[SI+2]         ;GET SECOND 2 BYTES
F709  8A 44 03                         MOV     AL,[SI+3]
F70C  E8 F714 R                        CALL    R56               ;BUILD LOW NIBBLE
F70F  88 56 00                         MOV     [BP],DL           ;STORE RESULT IN SAVE AREA
F712  45                               INC     BP                ;ADJUST POINTER
F713  C3                               RET
F714                           R55     ENDP
F714                           R56     PROC    NEAR
F714  B9 F000                          MOV     CX,0F000H         ; 4 BIT MASK TO TEST THE ENTRIES
F717  85 C1                    R57:    TEST    AX,CX             ; IS THIS SECTION BACKGROUND?
F719  74 01                            JZ      R58               ; IF ZERO, IT IS BACKGROUND
F71B  F9                               STC                       ; WASN'T, SO SET CARRY
F71C  D0 D2                    R58:    RCL     DL,1              ;MOVE THAT BIT INTO RESULT
F71E  D1 E9                            SHR     CX,1              ;MOVE MASK RIGH 4 BITS
F720  D1 E9                            SHR     CX,1
F722  D1 E9                            SHR     CX,1
F724  D1 E9                            SHR     CX,1
F726  73 EF                            JNC     R57               ;DO IT AGAIN IF MASK DID'T FALL OUT
F728  C3                               RET
F729                           R56     ENDP
```

```
                                ;---------------------------------------
                                ; V4_POSITION
                                ;  THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
                                ;  THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
                                ;  INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
                                ;  FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
                                ;  BE DOUBLED.
                                ; ENTRY -- NO REGISTERS,MEMORY LOCATION CURSOR_POSN IS USED
                                ; EXIT--
                                ;  AX CONTAINS OFFSET INTO REGEN BUFFER
                                ;---------------------------------------
F729                            R59     PROC    NEAR
F729  A1 0050 R                         MOV     AX,CURSOR_POSN  ; GET CURRENT CURSOR
F72C                            GRAPH_POSN      LABEL   NEAR
F72C  53                                PUSH    BX              ; SAVE REGISTER
F72D  8B D8                             MOV     BX,AX           ; SAVE A COPY OF CURRENT CURSOR
F72F  8A C4                             MOV     AL,AH           ; GET ROWS TO AL
F731  F6 26 004A R                      MUL     BYTE PTR CRT_COLS ; MULTIPLY BY BYTES/COLUMN
F735  80 3E 0049 R 09                   CMP     CRT_MODE,9      ; MODE USING 32K REGEN?
F73A  73 02                             JNC     R60             ; YES, JUMP
F73C  D1 E0                             SHL     AX,1            ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
F73E  D1 E0                     R60:    SHL     AX,1
F740  2A FF                             SUB     BH,BH           ; ISOLATE COLUMN VALUE
F742  03 C3                             ADD     AX,BX           ; DETERMINE OFFSET
F744  5B                                POP     BX              ; RECOVER POINTER
F745  C3                                RET                     ; ALL DONE
F746                            R59     ENDP
                                ;------------------------------------------------------------
                                ; LIGHT PEN
                                ;       THIS ROUTINE TESTS THE LIGHT PEN SWITCH AND THE LIGHT
                                ;       PEN TRIGGER. IF BOTH ARE SET, THE LOCATION OF THE LIGHT
                                ;       PEN IS DETERMINED. OTHERWISE, A RETURN WITH NO INFORMATION
                                ;       IS MADE.
                                ; ON EXIT:
                                ;       (AH) = 0 IF NO LIGHT PEN INFORMATION IS AVAILABLE
                                ;               BX,CX,DX ARE DESTROYED
                                ;       (AH) = 1 IF LIGHT PEN IS AVAILABLE
                                ;               (DH,DL) = ROW,COLUMN OF CURRENT LIGHT PEN POSITION
                                ;               (CH) = RASTER POSITION
                                ;               (BX) = BEST GUESS AT PIXEL HORIZONTAL POSITION
                                ;------------------------------------------------------------
                                        ASSUME  CS:CODE,DS:DATA
                                ;----- SUBTRACT_TABLE
F746                            V1      LABEL   BYTE
F746  03 03 05 05 03 03                 DB      3,3,5,5,3,3,3,0,2,3,4   ;
      03 00 02 03 04
F751                            READ_LPEN       PROC    NEAR
                                ;----- WAIT FOR LIGHT PEN TO BE DEPRESSED
F751  32 E4                             XOR     AH,AH           ; SET NO LIGHT PEN RETURN CODE
F753  BA 03DA                           MOV     DX,VGA_CTL      ;GET ADDRESS OF VGA CONTROL REG
F756  EC                                IN      AL,DX           ; GET STATUS REGISTER
F757  A8 04                             TEST    AL,4            ; TEST LIGHT PEN SWITCH
F759  74 03                             JZ      V7B
F75B  E9 F803 R                         JMP     V6              ; NOT SET, RETURN
                                ;----- NOW TEST FOR LIGHT PEN TRIGGER
F75E  A8 02                     V7B:    TEST    AL,2            ; TEST LIGHT PEN TRIGGER
F760  75 03                             JNZ     V7A             ; RETURN WITHOUT RESETTING TRIGGER
F762  E9 F80D R                         JMP     V7
                                ;----- TRIGGER HAS BEEN SET, READ THE VALUE IN
F765  B4 10                     V7A:    MOV     AH,16           ; LIGHT PEN REGISTERS ON 6845
                                ;----- INPUT REGS POINTED TO BY AH, AND CONVERT TO ROW COLUMN IN DX
F767  8B 16 0063 R                      MOV     DX,ADDR_6845    ; ADDRESS REGISTER FOR 6845
F76B  8A C4                             MOV     AL,AH           ; REGISTER TO READ
F76D  EE                                OUT     DX,AL           ; SET IT UP
F76E  42                                INC     DX              ; DATA REGISTER
F76F  EC                                IN      AL,DX           ; GET THE VALUE
F770  8A E8                             MOV     CH,AL           ; SAVE IN CX
F772  4A                                DEC     DX              ; ADDRESS REGISTER
F773  FE C4                             INC     AH
F775  8A C4                             MOV     AL,AH           ; SECOND DATA REGISTER
F777  EE                                OUT     DX,AL
F778  42                                INC     DX              ; POINT TO DATA REGISTER
F779  EC                                IN      AL,DX           ; GET SECOND DATA VALUE
F77A  8A E5                             MOV     AH,CH           ; AX HAS INPUT VALUE
                                ;----- AX HAS THE VALUE READ IN FROM THE 6845
F77C  8A 1E 0049 R                      MOV     BL,CRT_MODE
F780  2A FF                             SUB     BH,BH           ; MODE VALUE TO BX
F782  2E: 8A 9F F746 R                  MOV     BL,CS:V1[BX]    ; DETERMINE AMOUNT TO SUBTRACT
F787  2B C3                             SUB     AX,BX           ; TAKE IT AWAY
F789  3D 0FA0                           CMP     AX,4000         ; IN TOP OR BOTTOM BORDER?
F78C  72 02                             JB      V15             ; NO, OKAY
F78E  33 C0                             XOR     AX,AX           ; YES, SET TO ZERO
F790  8B 1E 004E R              V15:    MOV     BX,CRT_START
F794  D1 EB                             SHR     BX,1
F796  2B C3                             SUB     AX,BX           ; CONVERT TO CORRECT PAGE ORIGIN
F798  79 02                             JNS     V2              ; IF POSITIVE, DETERMINE MODE
F79A  2B C0                             SUB     AX,AX           ; <0 PLAYS AS 0
                                ;----- DETERMINE MODE OF OPERATION
F79C                            V2:                             ; DETERMINE_MODE
F79C  B1 03                             MOV     CL,3            ; SET *8 SHIFT COUNT
F79E  80 3E 0049 R 04                   CMP     CRT_MODE,4      ; DETERMINE IF GRAPHICS OR ALPHA
F7A3  72 4A                             JB      V4              ; ALPHA_PEN
                                ;----- GRAPHICS MODE
F7A5  B2 28                             MOV     DL,40           ; DIVISOR FOR GRAPHICS
F7A7  80 3E 0049 R 09                   CMP     CRT_MODE,9      ; USING 32K REGEN?
F7AC  72 02                             JB      V20             ; NO, JUMP
F7AE  B2 50                             MOV     DL,80           ; YES, SET RIGHT DIVSOR
F7B0  F6 F2                     V20:    DIV     DL              ; DETERMINE ROW(AL) AND COLUMN(AH)
                                                                ;  AL RANGE 0-99, AH RANGE 0-39
```

```
                                          ;----- DETERMINE GRAPHIC ROW POSITION
F7B2  8A E8                                       MOV     CH,AL               ; SAVE ROW VALUE IN CH
F7B4  02 ED                                       ADD     CH,CH               ; *2 FOR EVEN/ODD FIELD
F7B6  80 3E 0049 R 09                             CMP     CRT_MODE,9          ; USING 32K REGEN?
F7BB  72 06                                       JB      V21                 ; NO, JUMP
F7BD  D0 EC                                       SHR     AH,1                ; ADJUST ROW & COLUMN
F7BF  D0 E0                                       SHL     AL,1
F7C1  02 ED                                       ADD     CH,CH               ; *4 FOR 4 SCAN LINES
F7C3  8A DC                               V21:    MOV     BL,AH               ; COLUMN VALUE TO BX
F7C5  2A FF                                       SUB     BH,BH               ; MULTIPLY BY 8 FOR MEDIUM RES
F7C7  80 3E 0049 R 06                             CMP     CRT_MODE,6          ; DETERMINE MEDIUM OR HIGH RES
F7CC  72 15                                       JB      V3                  ; MODE 4 OR 5
F7CE  77 06                                       JA      V23                 ; MODE 8, 9, OR A
F7D0  B1 04                               V22:    MOV     CL,4                ; SHIFT VALUE FOR HIGH RES
F7D2  D0 E4                                       SAL     AH,1                ; COLUMN VALUE TIMES 2 FOR HIGH RES
F7D4  EB 0D                                       JMP     SHORT V3
F7D6  80 3E 0049 R 09                     V23:    CMP     CRT_MODE,9          ; CHECK MODE
F7DB  77 F3                                       JA      V22                 ; MODE A
F7DD  74 04                                       JE      V3                  ; MODE 9
F7DF  B1 02                                       MOV     CL,2                ; MODE 8 SHIFT VALUE
F7E1  D0 EC                                       SHR     AH,1
F7E3                                      V3:                                 ; NOT_HIGH_RES
F7E3  D3 E3                                       SHL     BX,CL               ; MULTIPLY *16 FOR HIGH RES
                                          ;----- DETERMINE ALPHA CHAR POSITION
F7E5  8A D4                                       MOV     DL,AH               ; COLUMN VALUE FOR RETURN
F7E7  8A F0                                       MOV     DH,AL               ; ROW VALUE
F7E9  D0 EE                                       SHR     DH,1                ; DIVIDE BY 4
F7EB  D0 EE                                       SHR     DH,1                ;  FOR VALUE IN 0-24 RANGE
F7ED  EB 12                                       JMP     SHORT V5            ; LIGHT_PEN_RETURN_SET
                                          ;----- ALPHA MODE ON LIGHT PEN
F7EF                                      V4:                                 ; ALPHA_PEN
F7EF  F6 36 004A R                                DIV     BYTE PTR CRT_COLS ; DETERMINE ROW,COLUMN VALUE
F7F3  8A F0                                       MOV     DH,AL               ; ROWS TO DH
F7F5  8A D4                                       MOV     DL,AH               ; COLS TO DL
F7F7  D2 E0                                       SAL     AL,CL               ; MULTIPLY ROWS * 8
F7F9  8A E8                                       MOV     CH,AL               ; GET RASTER VALUE TO RETURN REG
F7FB  8A DC                                       MOV     BL,AH               ; COLUMN VALUE
F7FD  32 FF                                       XOR     BH,BH               ;  TO BX
F7FF  D3 E3                                       SAL     BX,CL
F801                                      V5:                                 ; LIGHT_PEN_RETURN_SET
F801  B4 01                                       MOV     AH,1                ; INDICATE EVERYTHING SET
F803                                      V6:                                 ; LIGHT_PEN_RETURN
F803  52                                          PUSH    DX                  ; SAVE RETURN VALUE (IN CASE)
F804  8B 16 0063 R                                MOV     DX,ADDR_6845        ; GET BASE ADDRESS
F808  83 C2 07                                    ADD     DX,7                ; POINT TO RESET PARM
F80B  EE                                          OUT     DX,AL               ; ADDRESS, NOT DATA, IS IMPORTANT
F80C  5A                                          POP     DX                  ; RECOVER VALUE
F80D                                      V7:                                 ; RETURN_NO_RESET
F80D  5F                                          POP     DI
F80E  5E                                          POP     SI
F80F  1F                                          POP     DS                  ; DISCARD SAVED BX,CX,DX
F810  1F                                          POP     DS
F811  1F                                          POP     DS
F812  1F                                          POP     DS
F813  07                                          POP     ES
F814  CF                                          IRET
F815                                      READ_LPEN       ENDP
                                          ;--------------------------------------------------------
                                          ; TEMPORARY INTERRUPT SERVICE ROUTINE                   :
                                          ;         1. THIS ROUTINE IS ALSO LEFT IN PLACE AFTER THE :
                                          ;         POWER ON DIAGNOSTICS TO SERVICE UNUSED         :
                                          ;         INTERRUPT VECTORS. LOCATION 'INTR_FLAG' WILL    :
                                          ;         CONTAIN EITHER: 1. LEVEL OF HARDWARE INT. THAT  :
                                          ;         CAUSED CODE TO BE EXEC.                         :
                                          ;         2. 'FF' FOR NON-HARDWARE INTERRUPTS THAT WERE   :
                                          ;         EXECUTED ACCIDENTLY.                            :
                                          ;--------------------------------------------------------
F815                                      D11     PROC    NEAR
                                                  ASSUME  DS:DATA
F815  1E                                          PUSH    DS
F816  50                                          PUSH    AX                  ; SAVE REG AX CONTENTS
F817  E8 138B R                                   CALL    DDS
F81A  B0 0B                                       MOV     AL,0BH              ; READ IN-SERVICE REG
F81C  E6 20                                       OUT     INTA00,AL           ; (FIND OUT WHAT LEVEL BEING
F81E  90                                          NOP                         ; SERVICED)
F81F  E4 20                                       IN      AL,INTA00           ; GET LEVEL
F821  8A E0                                       MOV     AH,AL               ; SAVE IT
F823  0A C4                                       OR      AL,AH               ; 00? (NO HARDWARE ISR ACTIVE)
F825  75 04                                       JNZ     HW_INT
F827  B4 FF                                       MOV     AH,0FFH
F829  EB 0A                                       JMP     SHORT SET_INTR_FLAG ; SET FLAG TO FF IF NON-HDWARE
F82B  E4 21                               HW_INT: IN      AL,INTA01           ; GET MASK VALUE
F82D  0A C4                                       OR      AL,AH               ; MASK OFF LVL BEING SERVICED
F82F  E6 21                                       OUT     INTA01,AL
F831  B0 20                                       MOV     AL,EOI
F833  E6 20                                       OUT     INTA00,AL
F835                                      SET_INTR_FLAG:
F835  88 26 0084 R                                MOV     INTR_FLAG,AH        ; SET FLAG
F839  58                                          POP     AX                  ; RESTORE REG AX CONTENTS
F83A  1F                                          POP     DS
F83B  FB                                          STI                         ; INTERRUPTS BACK ON
F83C                                      DUMMY_RETURN:                       ; NEED IRET FOR VECTOR TABLE
F83C  CF                                          IRET
F83D                                      D11     ENDP
```

```
                              ;--- INT 12 -------------------------------
                              ; MEMORY_SIZE_DETERMINE
                              ; INPUT
                              ;       NO REGISTERS
                              ;       THE MEMORY_SIZE VARIABLE IS SET DURING POWER ON DIAGNOSTICS
                              ; OUTPUT
                              ;       (AX) = NUMBER OF CONTIGUOUS 1K BLOCKS OF MEMORY
                              ;-------------------------------------------
                                      ASSUME  CS:CODE,DS:DATA
F841                                  ORG     0F841H
F841                          MEMORY_SIZE_DETERMINE   PROC    FAR
F841  FB                              STI                     ; INTERRUPTS BACK ON
F842  1E                              PUSH    DS              ; SAVE SEGMENT
F843  B8  ---- R                      MOV     AX,DATA         ; ESTABLISH ADDRESSING
F846  8E D8                           MOV     DS,AX
F848  A1 0013 R                       MOV     AX,MEMORY_SIZE  ; GET VALUE
F84B  1F                              POP     DS              ; RECOVER SEGMENT
F84C  CF                              IRET                    ; RETURN TO CALLER
F84D                          MEMORY_SIZE_DETERMINE   ENDP
                              ;--- INT 11 ---------------------------------
                              ; EQUIPMENT DETERMINATION
                              ;       THIS ROUTINE ATTEMPTS TO DETERMINE WHAT OPTIONAL
                              ;       DEVICES ARE ATTACHED TO THE SYSTEM.
                              ; INPUT
                              ;       NO REGISTERS
                              ;       THE EQUIP_FLAG VARIABLE IS SET DURING THE POWER ON
                              ;        DIAGNOSTICS USING THE FOLLOWING HARDWARE ASSUMPTIONS:
                              ;       PORT 62 (0->3) = LOW ORDER BYTE OF EQUIPMENT
                              ;       PORT 3FA = INTERRUPT ID REGISTER OF 8250
                              ;               BITS 7-3 ARE ALWAYS 0
                              ;       PORT 378 = OUTPUT PORT OF PRINTER -- 8255 PORT THAT
                              ;               CAN BE READ AS WELL AS WRITTEN
                              ;  OUTPUT
                              ;       (AX) IS SET, BIT SIGNIFICANT, TO INDICATE ATTACHED I/O
                              ;       BIT 15,14 = NUMBER OF PRINTERS ATTACHED
                              ;       BIT 13 = 1 = SERIAL PRINTER ATTACHED
                              ;       BIT 12 = GAME I/O ATTACHED
                              ;       BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
                              ;       BIT 8  0 = DMA CHIP PRESENT ON SYSTEM, 1 = NO DMA ON SYSTEM
                              ;       BIT 7,6 = NUMBER OF DISKETTE DRIVES
                              ;               00=1, 01=2, 10=3, 11=4 ONLY IF BIT 0 = 1
                              ;       BIT 5,4 = INITIAL VIDEO MODE
                              ;                       00 - UNUSED
                              ;                       01 - 40X25 BW USING COLOR CARD
                              ;                       10 - 80X25 BW USING COLOR CARD
                              ;                       11 - 80X25 BW USING BW CARD
                              ;       BIT 3,2 = PLANAR RAM SIZE (10=48K,11=64K)
                              ;       BIT 1 NOT USED
                              ;       BIT 0 = 1 (IPL DISKETTE INSTALLED)
                              ;       NO OTHER REGISTERS AFFECTED
                              ;-----------------------------------------------
                                      ASSUME  CS:CODE,DS:DATA
F84D                                  ORG      0F84DH
F84D                          EQUIPMENT       PROC    FAR
F84D  FB                              STI                     ; INTERRUPTS BACK ON
F84E  1E                              PUSH    DS              ; SAVE SEGMENT REGISTER
F84F  B8  ---- R                      MOV     AX,DATA         ; ESTABLISH ADDRESSING
F852  8E D8                           MOV     DS,AX
F854  A1 0010 R                       MOV     AX,EQUIP_FLAG   ; GET THE CURRENT SETTINGS
F857  1F                              POP     DS              ; RECOVER SEGMENT
F858  CF                              IRET                    ; RETURN TO CALLER
F859                          EQUIPMENT       ENDP
                              ;--- INT 15 ------------------------------------------------
                              ; CASSETTE I/O
                              ;       (AH) = 0  TURN CASSETTE MOTOR ON
                              ;       (AH) = 1  TURN CASSETTE MOTOR OFF
                              ;       (AH) = 2  READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
                              ;               (ES,BX) = POINTER TO DATA BUFFER
                              ;               (CX) = COUNT OF BYTES TO READ
                              ; ON EXIT
                              ;       (ES,BX) = POINTER TO LAST BYTE READ + 1
                              ;       (DX) = COUNT OF BYTES ACTUALLY READ
                              ;       (CY) = 0 IF NO ERROR OCCURRED
                              ;            = 1 IF ERROR OCCURRED
                              ;       (AH) = ERROR RETURN IF (CY)= 1
                              ;               = 01 IF CRC ERROR WAS DETECTED
                              ;               = 02 IF DATA TRANSITIONS ARE LOST
                              ;               = 04 IF NO DATA WAS FOUND
                              ;       (AH) = 3 WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE
                              ;               (ES,BX) = POINTER TO DATA BUFFER
                              ;               (CX) = COUNT OF BYTES TO WRITE
                              ; ON EXIT
                              ;       (EX,BX) = POINTER TO LAST BYTE WRITTEN + 1
                              ;       (CX) = 0
                              ;       (AH) = ANY OTHER THAN ABOVE VALUES CAUSES (CY)= 1
                              ;               AND (AH)= 80 TO BE RETURNED (INVALID COMMAND).
                              ;-------------------------------------------------------
                                      ASSUME  DS:DATA, ES:NOTHING,SS:NOTHING,CS:CODE
F859                                  ORG     0F859H
F859                          CASSETTE_IO     PROC    FAR
F859  FB                              STI                     ; INTERRUPTS BACK ON
F85A  1E                              PUSH    DS              ; ESTABLISH ADDRESSING TO DATA
F85B  E8 138B R                       CALL    DDS
F85E  80 26 0071 R 7F                 AND     BIOS_BREAK, 7FH ; MAKE SURE BREAK FLAG IS OFF
F863  E8 F86A R                       CALL    W1              ; CASSETTE_IO_CONT
F866  1F                              POP     DS
F867  CA 0002                         RET     2               ; INTERRUPT RETURN
F86A                          CASSETTE_IO     ENDP
F86A                          W1      PROC    NEAR
```

```
                              ;--------------------------------------------------
                              ; PURPOSE:
                              ;  TO CALL APPROPRIATE ROUTINE DEPENDING ON REG AH
                              ;  AH           ROUTINE
                              ;--------------------------------------------------
                              ;  0            MOTOR ON
                              ;  1            MOTOR OFF
                              ;  2            READ CASSETTE BLOCK
                              ;  3            WRITE CASSETTE BLOCK
                              ;--------------------------------------------------
F86A  0A E4                          OR      AH,AH           ; TURN ON MOTOR?
F86C  74 13                          JZ      MOTOR_ON        ; YES, DO IT
F86E  FE CC                          DEC     AH              ; TURN OFF MOTOR?
F870  74 18                          JZ      MOTOR_OFF       ; YES, DO IT
F872  FE CC                          DEC     AH              ; READ CASSETTE BLOCK?
F874  74 1A                          JZ      READ_BLOCK      ; YES, DO IT
F876  FE CC                          DEC     AH              ; WRITE CASSETTE BLOCK?
F878  75 03                          JNZ     W2              ;  NOT_DEFINED
F87A  E9 F997 R                      JMP     WRITE_BLOCK     ; YES, DO IT
F87D                          W2:                            ; COMMAND NOT DEFINED
F87D  B4 80                          MOV     AH,080H         ; ERROR, UNDEFINED OPERATION
F87F  F9                             STC                     ; ERROR FLAG
F880  C3                             RET
F881                          W1     ENDP
F881                          MOTOR_ON       PROC    NEAR
                              ;-----------------------------------
                              ; PURPOSE:
                              ;       TO TURN ON CASSETTE MOTOR
                              ;-----------------------------------
F881  E4 61                          IN      AL,PORT_B       ; READ CASSETTE OUTPUT
F883  24 F7                          AND     AL,NOT 08H      ; CLEAR BIT TO TURN ON MOTOR
F885  E6 61                   W3:    OUT     PORT_B,AL       ; WRITE IT OUT
F887  2A E4                          SUB     AH,AH           ; CLEAR AH
F889  C3                             RET
F88A                          MOTOR_ON       ENDP
F88A                          MOTOR_OFF      PROC    NEAR
                              ;-----------------------------------
                              ; PURPOSE:                          :
                              ;       TO TURN CASSETTE MOTOR OFF  :
                              ;-----------------------------------
F88A  E4 61                          IN      AL,PORT_B       ; READ CASSETTE OUTPUT
F88C  0C 08                          OR      AL,08H          ; SET BIT TO TURN OFF
F88E  EB F5                          JMP     W3              ; WRITE IT, CLEAR ERROR, RETURN
F890                          MOTOR_OFF      ENDP
F890                          READ_BLOCK     PROC    NEAR
                              ;-----------------------------------------------------------
                              ; PURPOSE:
                              ;       TO READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
                              ;
                              ; ON ENTRY:
                              ;       ES IS SEGMENT FOR MEMORY BUFFER (FOR COMPACT CODE)
                              ;       BX POINTS TO START OF MEMORY BUFFER
                              ;       CX CONTAINS NUMBER OF BYTES TO READ
                              ; ON EXIT:
                              ;       BX POINTS 1 BYTE PAST LAST BYTE PUT IN MEM
                              ;       CX CONTAINS DECREMENTED BYTE COUNT
                              ;       DX CONTAINS NUMBER OF BYTES ACTUALLY READ
                              ;
                              ;       CARRY FLAG IS CLEAR IF NO ERROR DETECTED
                              ;       CARRY FLAG IS SET IF CRC ERROR DETECTED
                              ;-----------------------------------------------------------
F890  53                             PUSH    BX              ; SAVE BX
F891  51                             PUSH    CX              ; SAVE CX
F892  56                             PUSH    SI              ; SAVE SI
F893  BE 0007                        MOV     SI, 7           ; SET UP RETRY COUNT FOR LEADER
F896  E8 FA50 R                      CALL    BEGIN_OP        ; BEGIN BY STARTING MOTOR
F899                          W4:                            ; SEARCH FOR LEADER
F899  E4 62                          IN      AL,PORT_C       ; GET INITIAL VALUE
F89B  24 10                          AND     AL,010H         ; MASK OFF EXTRANEOUS BITS
F89D  A2 006B R                      MOV     LAST_VAL,AL     ; SAVE IN LOC LAST_VAL
F8A0  BA 3F7A                        MOV     DX,16250        ; # OF TRANSITIONS TO LOOK FOR
F8A3                          W5:                            ; WAIT_FOR_EDGE
F8A3  F6 06 0071 R 80                TEST    BIOS_BREAK, 80H ; CHECK FOR BREAK KEY
F8A8  75 03                          JNZ     W6A             ; JUMP IF NO BREAK KEY
                                                             ; JUMP IF BREAK KEY HIT
F8AA  4A                             DEC     DX
F8AB  75 03                          JNZ     W7              ; JUMP IF BEGINNING OF LEADER
F8AD  E9 F92F R               W6A:   JMP     W17             ; JUMP IF NO LEADER FOUND
F8B0  E8 F96F R               W7:    CALL    READ_HALF_BIT   ; IGNORE FIRST EDGE
F8B3  E3 EE                          JCXZ    W5              ; JUMP IF NO EDGE DETECTED
F8B5  BA 0378                        MOV     DX,0378H        ; CHECK FOR HALF BITS
F8B8  B9 0200                        MOV     CX,200H         ; MUST HAVE AT LEAST THIS MANY ONE
                                                             ; SIZE PULSES BEFORE CHCKNG FOR
                                                             ; SYNC BIT (0)
F8BB  FA                             CLI                     ; DISABLE INTERRUPTS
F8BC                          W8:                            ; SEARCH-LDR
F8BC  F6 06 0071 R 80                TEST    BIOS_BREAK, 80H ; CHECK FOR BREAK KEY
F8C1  75 6C                          JNZ     W17             ; JUMP IF BREAK KEY HIT
F8C3  51                             PUSH    CX              ; SAVE REG CX
F8C4  E8 F96F R                      CALL    READ_HALF_BIT   ; GET PULSE WIDTH
F8C7  0B C9                          OR      CX, CX          ; CHECK FOR TRANSITION
F8C9  59                             POP     CX              ; RESTORE ONE BIT COUNTER
F8CA  74 CD                          JZ      W4              ; JUMP IF NO TRANSITION
F8CC  3B D3                          CMP     DX,BX           ; CHECK PULSE WIDTH
F8CE  E3 04                          JCXZ    W9              ; IF CX=0 THEN WE CAN LOOK
                                                             ; FOR SYNC BIT (0)
F8D0  73 C7                          JNC     W4              ; JUMP IF ZERO BIT (NOT GOOD
                                                             ; LEADER)
F8D2  E2 E8                          LOOP    W8              ; DEC CX AND READ ANOTHER HALF ONE
                                                             ; BIT
F8D4                          W9:                            ; FIND-SYNC
F8D4  72 E6                          JC      W8              ; JUMP IF ONE BIT (STILL LEADER)
```

```
                               ;----- A SYNCH BIT HAS BEEN FOUND.  READ SYN CHARACTER:
F8D6  E8 F96F R                        CALL    READ_HALF_BIT     ; SKIP OTHER HALF OF SYNC BIT (0)
F8D9  E8 F941 R                        CALL    READ_BYTE         ; READ SYNC BYTE
F8DC  3C 16                            CMP     AL, 16H           ; SYNCHRONIZATION CHARACTER
F8DE  75 49                            JNE     W16               ; JUMP IF BAD LEADER FOUND.
                               ;----- GOOD CRC SO READ DATA BLOCK(S)
F8E0  5E                               POP     SI                ; RESTORE REGS
F8E1  59                               POP     CX
F8E2  5B                               POP     BX
                               ;--------------------------------------------------------------
                               ; READ 1 OR MORE 256 BYTE BLOCKS FROM CASSETTE
                               ; ON ENTRY:
                               ;       ES IS SEGMENT FOR MEMORY BUFFER (FOR COMPACT CODE)
                               ;       BX POINTS TO START OF MEMORY BUFFER
                               ;       CX CONTAINS NUMBER OF BYTES TO READ
                               ; ON EXIT:
                               ;       BX POINTS 1 BYTE PAST LAST BYTE PUT IN MEM
                               ;       CX CONTAINS DECREMENTED BYTE COUNT
                               ;       DX CONTAINS NUMBER OF BYTES ACTUALLY READ
                               ;--------------------------------------------------------------
F8E3  51                               PUSH    CX                ; SAVE BYTE COUNT
F8E4                           W10:                              ; COME HERE BEFORE EACH
                                                                 ; 256 BYTE BLOCK IS READ
F8E4  C7 06 0069 R FFFF                MOV     CRC_REG,0FFFFH    ; INIT CRC REG
F8EA  BA 0100                          MOV     DX,256            ; SET DX TO DATA BLOCK SIZE
F8ED                           W11:                              ; RD_BLK
F8ED  F6 06 0071 R 80                  TEST    BIOS_BREAK, 80H   ; CHECK FOR BREAK KEY
F8F2  75 23                            JNZ     W13               ; JUMP IF BREAK KEY HIT
F8F4  E8 F941 R                        CALL    READ_BYTE         ; READ BYTE FROM CASSETTE
F8F7  72 1E                            JC      W13               ; CY SET INDICATES NO DATA
                                                                 ; TRANSITIONS
F8F9  E3 05                            JCXZ    W12               ; IF WE'VE ALREADY REACHED
                                                                 ; END OF MEMORY BUFFER
                                                                 ; SKIP REST OF BLOCK
F8FB  26: 88 07                        MOV     ES:[BX],AL        ; STORE DATA BYTE AT BYTE PTR
F8FE  43                               INC     BX                ; INC BUFFER PTR
F8FF  49                               DEC     CX                ; DEC BYTE COUNTER
F900                           W12:            ; LOOP UNTIL DATA BLOCK HAS BEEN READ FROM CASSETTE
F900  4A                               DEC     DX                ; DEC BLOCK CNT
F901  7F EA                            JG      W11               ; RD_BLK
F903  E8 F941 R                        CALL    READ_BYTE         ; NOW READ TWO CRC BYTES
F906  E8 F941 R                        CALL    READ_BYTE
F909  2A E4                            SUB     AH,AH             ; CLEAR AH
F90B  81 3E 0069 R 1D0F                CMP     CRC_REG,1D0FH     ; IS THE CRC CORRECT?
F911  75 06                            JNE     W14               ; IF NOT EQUAL CRC IS BAD
F913  E3 06                            JCXZ    W15               ; IF BYTE COUNT IS ZERO
                                                                 ; THEN WE HAVE READ ENOUGH
                                                                 ; SO WE WILL EXIT
F915  EB CD                            JMP     W10               ; STILL MORE, SO READ ANOTHER BLOCK
F917                           W13:                              ; MISSING-DATA
                                                                 ; NO DATA TRANSITIONS SO
F917  B4 01                            MOV     AH,01H            ; SET AH=02 TO INDICATE
                                                                 ; DATA TIMEOUT
F919                           W14:                              ; BAD-CRC
F919  FE C4                            INC     AH                ; EXIT EARLY ON ERROR
                                                                 ; SET AH=01 TO INDICATE CRC ERROR
F91B                           W15:                              ; RD-BLK-EX
F91B  5A                               POP     DX                ; CALCULATE COUNT OF
F91C  2B D1                            SUB     DX,CX             ; DATA BYTES ACTUALLY READ
                                                                 ; RETURN COUNT IN REG DX
F91E  50                               PUSH    AX                ; SAVE AX (RET CODE)
F91F  F6 C4 90                         TEST    AH, 90H           ; CHECK FOR ERRORS
F922  75 13                            JNZ     W18               ; JUMP IF ERROR DETECTED
F924  E8 F941 R                        CALL    READ_BYTE         ; READ TRAILER
F927  EB 0E                            JMP     SHORT W18         ; SKIP TO TURN OFF MOTOR
F929                           W16:                              ; BAD-LEADER
F929  4E                               DEC     SI                ; CHECK RETRIES
F92A  74 03                            JZ      W17               ; JUMP IF TOO MANY RETRIES
F92C  E9 F899 R                        JMP     W4                ; JUMP IF NOT TOO MANY RETRIES
F92F                           W17:                              ; NO VALID DATA FOUND
                               ;----- NO DATA FROM CASSETTE ERROR, I.E. TIMEOUT
F92F  5E                               POP     SI                ; RESTORE REGS
F930  59                               POP     CX                ; RESTORE REGS
F931  5B                               POP     BX
F932  2B D2                            SUB     DX,DX             ; ZERO NUMBER OF BYTES READ
F934  B4 04                            MOV     AH,04H            ; TIME OUT ERROR (NO LEADER)
F936  50                               PUSH    AX
F937                           W18:                              ; MOT-OFF
F937  FB                               STI                       ; REENABLE INTERRUPTS
F938  E8 F88A R                        CALL    MOTOR_OFF         ; TURN OFF MOTOR
F93B  58                               POP     AX                ; RESTORE RETURN CODE
F93C  80 FC 01                         CMP     AH,01H            ; SET CARRY IF ERROR (AH>0)
F93F  F5                               CMC
F940  C3                               RET                       ; FINISHED
F941                           READ_BLOCK      ENDP
```

```
                          ;-----------------------------------------
                          ; PURPOSE:
                          ;       TO READ A BYTE FROM CASSETTE
                          ; ON EXIT
                          ;       REG AL CONTAINS READ DATA BYTE
                          ;-----------------------------------------
F941                      READ_BYTE       PROC    NEAR
F941  53                          PUSH    BX                ; SAVE REGS BX,CX
F942  51                          PUSH    CX
F943  B1 08                       MOV     CL,8H             ; SET BIT COUNTER FOR 8 BITS
F945                      W19:                              ; BYTE-ASM
F945  51                          PUSH    CX                ; SAVE CX
                          ;--------------------------------
                          ; READ DATA BIT FROM CASSETTE
                          ;--------------------------------
F946  E8 F96F R                   CALL    READ_HALF_BIT     ; READ ONE PULSE
F949  E3 20                       JCXZ    W21               ; IF CX=0 THEN TIMEOUT
                                                            ; BECAUSE OF NO DATA TRANSITIONS
F94B  53                          PUSH    BX                ; SAVE 1ST HALF BIT'S
                                                            ; PULSE WIDTH (IN BX)
F94C  E8 F96F R                   CALL    READ_HALF_BIT     ; READ COMPLEMENTARY PULSE
F94F  58                          POP     AX                ; COMPUTE DATA BIT
F950  E3 19                       JCXZ    W21               ; IF CX=0 THEN TIMEOUT DUE TO
                                                            ; NO DATA TRANSITIONS
F952  03 D8                       ADD     BX,AX             ; PERIOD
F954  81 FB 06F0                  CMP     BX, 06F0H         ; CHECK FOR ZERO BIT
F958  F5                          CMC                       ; CARRY IS SET IF ONE BIT
F959  9F                          LAHF                      ; SAVE CARRY IN AH
F95A  59                          POP     CX                ; RESTORE CX
                                                            ; NOTE:
                                                            ; MS BIT OF BYTE IS READ FIRST.
                                                            ; REG CH IS SHIFTED LEFT WITH
                                                            ; CARRY BEING INSERTED INTO LS
                                                            ; BIT OF CH.
                                                            ; AFTER ALL 8 BITS HAVE BEEN
                                                            ; READ, THE MS BIT OF THE DATA
                                                            ; BYTE WILL BE IN THE MS BIT OF
                                                            ; REG CH
F95B  D0 D5                       RCL     CH,1              ; ROTATE REG CH LEFT WITH CARRY TO
                                                            ;   LS BIT OF REG CH
F95D  9E                          SAHF                      ; RESTORE CARRY FOR CRC ROUTINE
F95E  E8 FA3C R                   CALL    CRC_GEN           ; GENERATE CRC FOR BIT
F961  FE C9                       DEC     CL                ; LOOP TILL ALL 8 BITS OF DATA
                                                            ; ASSEMBLED IN REG CH
F963  75 E0                       JNZ     W19               ; BYTE_ASM
F965  8A C5                       MOV     AL,CH             ; RETURN DATA BYTE IN REG AL
F967  F8                          CLC
F968                      W20:                              ; RD-BYT-EX
F968  59                          POP     CX                ; RESTORE REGS CX,BX
F969  5B                          POP     BX
F96A  C3                          RET                       ; FINISHED
F96B                      W21:                              ; NO-DATA
F96B  59                          POP     CX                ; RESTORE CX
F96C  F9                          STC                       ; INDICATE ERROR
F96D  EB F9                       JMP     W20               ; RD_BYT_EX
F96F                      READ_BYTE       ENDP
                          ;-----------------------------------------------
                          ; PURPOSE:
                          ;       TO COMPUTE TIME TILL NEXT DATA
                          ;       TRANSITION (EDGE)
                          ; ON ENTRY:
                          ;       EDGE_CNT CONTAINS LAST EDGE COUNT
                          ; ON EXIT:
                          ;       AX CONTAINS OLD LAST EDGE COUNT
                          ;       BX CONTAINS PULSE WIDTH (HALF BIT)
                          ;-----------------------------------------------
F96F                      READ_HALF_BIT   PROC    NEAR
F96F  B9 0064                     MOV     CX, 100           ; SET TIME TO WAIT FOR BIT
F972  8A 26 006B R                MOV     AH,LAST_VAL       ; GET PRESENT INPUT VALUE
F976                      W22:                              ; RD-H-BIT
F976  E4 62                       IN      AL,PORT_C         ; INPUT DATA BIT
F978  24 10                       AND     AL,010H           ; MASK OFF EXTRANEOUS BITS
F97A  3A C4                       CMP     AL,AH             ; SAME AS BEFORE?
F97C  E1 F8                       LOOPE   W22               ; LOOP TILL IT CHANGES
F97E  A2 006B R                   MOV     LAST_VAL,AL       ; UPDATE LAST_VAL WITH NEW VALUE
F981  B0 40                       MOV     AL,40H            ; READ TIMER'S COUNTER COMMAND
F983  E6 43                       OUT     TIM_CTL,AL        ; LATCH COUNTER
F985  8B 1E 0067 R                MOV     BX,EDGE_CNT       ; BX GETS LAST EDGE COUNT
F989  E4 41                       IN      AL,TIMER+1        ; GET LS BYTE
F98B  8A E0                       MOV     AH,AL             ; SAVE IN AH
F98D  E4 41                       IN      AL,TIMER+1        ; GET MS BYTE
F98F  86 C4                       XCHG    AL,AH             ; XCHG AL,AH
F991  2B D8                       SUB     BX,AX             ; SET BX EQUAL TO HALF BIT PERIOD
F993  A3 0067 R                   MOV     EDGE_CNT,AX       ; UPDATE EDGE COUNT;
F996  C3                          RET
F997                      READ_HALF_BIT   ENDP
```

```
                                   ;------------------------------------------------------------
                                   ; PURPOSE
                                   ;       WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE.
                                   ;       THE DATA IS PADDED TO FILL OUT THE LAST 256 BYTE BLOCK.
                                   ; ON ENTRY:
                                   ;       BX POINTS TO MEMORY BUFFER ADDRESS
                                   ;       CX CONTAINS NUMBER OF BYTES TO WRITE
                                   ; ON EXIT:
                                   ;       BX POINTS 1 BYTE PAST LAST BYTE WRITTEN TO CASSETTE
                                   ;       CX IS ZERO
                                   ;------------------------------------------------------------
F997                               WRITE_BLOCK     PROC    NEAR
F997  53                                   PUSH    BX
F998  51                                   PUSH    CX
F999  E4 61                                IN      AL,PORT_B        ; DISABLE SPEAKER
F99B  24 FD                                AND     AL,NOT 02H
F99D  0C 01                                OR      AL, 01H          ; ENABLE TIMER
F99F  E6 61                                OUT     PORT_B,AL
F9A1  B0 B6                                MOV     AL,0B6H          ; SET UP TIMER - MODE 3 SQUARE WAVE
F9A3  E6 43                                OUT     TIM_CTL,AL
F9A5  E8 FA50 R                            CALL    BEGIN_OP         ; START MOTOR AND DELAY
F9A8  B8 04A0                              MOV     AX,1184          ; SET NORMAL BIT SIZE
F9AB  E8 FA35 R                            CALL    W31              ; SET_TIMER
F9AE  B9 0800                              MOV     CX,0800H         ; SET CX FOR LEADER BYTE COUNT
F9B1                               W23:                             ; WRITE LEADER
F9B1  F9                                   STC                      ; WRITE ONE BITS
F9B2  E8 FA1F R                            CALL    WRITE_BIT
F9B5  E2 FA                                LOOP    W23              ; LOOP 'TIL LEADER IS WRITTEN
F9B7  FA                                   CLI                      ; DISABLE INTS.
F9B8  F8                                   CLC                      ; WRITE SYNC BIT (0)
F9B9  E8 FA1F R                            CALL    WRITE_BIT
F9BC  59                                   POP     CX               ; RESTORE REGS CX,BX
F9BD  5B                                   POP     BX
F9BE  B0 16                                MOV     AL, 16H          ; WRITE SYNC CHARACTER
F9C0  E8 FA08 R                            CALL    WRITE_BYTE
                                   ;------------------------------------------------------------
                                   ; PURPOSE
                                   ;       WRITE 1 OR MORE 256 BYTE BLOCKS TO CASSETTE
                                   ; ON ENTRY:
                                   ;       BX POINTS TO MEMORY BUFFER ADDRESS
                                   ;       CONTAINS NUMBER OF BYTES TO WRITE
                                   ; ON EXIT:
                                   ;       BX POINTS 1 BYTE PAST LAST BYTE WRITTEN TO CASSETTE
                                   ;       CX IS ZERO
                                   ;------------------------------------------------------------
F9C3                               WR_BLOCK:
F9C3  C7 06 0069 R FFFF                    MOV     CRC_REG,0FFFFH   ; INIT CRC
F9C9  BA 0100                              MOV     DX,256           ; FOR 256 BYTES
F9CC                               W24:                             ; WR-BLK
F9CC  26: 8A 07                            MOV     AL,ES:[BX]       ; READ BYTE FROM MEM
F9CF  E8 FA08 R                            CALL    WRITE_BYTE       ; WRITE IT TO CASSETTE
F9D2  E3 02                                JCXZ    W25              ; UNLESS CX=0, ADVANCE PTRS & DEC
                                                                    ; COUNT
F9D4  43                                   INC     BX               ; INC BUFFER POINTER
F9D5  49                                   DEC     CX               ; DEC BYTE COUNTER
F9D6                               W25:                             ; SKIP-ADV
F9D6  4A                                   DEC     DX               ; DEC BLOCK CNT
F9D7  7F F3                                JG      W24              ; LOOP TILL 256 BYTE BLOCK
                                                                    ;  IS WRITTEN TO TAPE
                                   ;------------------------------------------------------------
                                   ; WRITE CRC
                                   ;       WRITE 1'S COMPLEMENT OF CRC REG TO CASSETTE
                                   ;       WHICH IS CHECKED FOR CORRECTNESS WHEN THE BLOCK IS READ
                                   ; REG AX IS MODIFIED
                                   ;------------------------------------------------------------
F9D9  A1 0069 R                            MOV     AX,CRC_REG       ; WRITE THE ONE'S COMPLEMENT OF THE
                                                                    ;  TWO BYTE CRC TO TAPE
F9DC  F7 D0                                NOT     AX               ; FOR 1'S COMPLEMENT
F9DE  50                                   PUSH    AX               ; SAVE IT
F9DF  86 E0                                XCHG    AH,AL            ; WRITE MS BYTE FIRST
F9E1  E8 FA08 R                            CALL    WRITE_BYTE       ; WRITE IT
F9E4  58                                   POP     AX               ; GET IT BACK
F9E5  E8 FA08 R                            CALL    WRITE_BYTE       ; NOW WRITE LS BYTE
F9E8  0B C9                                OR      CX,CX            ; IS BYTE COUNT EXHAUSTED?
F9EA  75 D7                                JNZ     WR_BLOCK         ; JUMP IF NOT DONE YET
F9EC  51                                   PUSH    CX               ; SAVE REG CX
F9ED  FB                                   STI                      ; RE-ENABLE INTERUPTS
F9EE  B9 0020                              MOV     CX, 32           , WRITE OUT TRAILER BITS
F9F1                               W26:                             ; TRAIL-LOOP
F9F1  F9                                   STC
F9F2  E8 FA1F R                            CALL    WRITE_BIT
F9F5  E2 FA                                LOOP    W26              ; WRITE UNTIL TRAILER WRITTEN
F9F7  59                                   POP     CX               ; RESTORE REG CX
F9F8  B0 B0                                MOV     AL, 0B0H         ; TURN TIMER2 OFF
F9FA  E6 43                                OUT     TIM_CTL, AL
F9FC  B8 0001                              MOV     AX, 1
F9FF  E8 FA35 R                            CALL    W31              ; SET_TIMER
FA02  E8 F88A R                            CALL    MOTOR_OFF        ; TURN MOTOR OFF
FA05  2B C0                                SUB     AX,AX            ; NO ERRORS REPORTED ON WRITE OP
FA07  C3                                   RET                      ; FINISHED
FA08                               WRITE_BLOCK     ENDP
```

```
                                ;--------------------------------
                                ; WRITE A BYTE TO CASSETTE.
                                ; BYTE TO WRITE IS IN REG AL.
                                ;--------------------------------
FA08                            WRITE_BYTE      PROC    NEAR
FA08  51                                PUSH    CX              ; SAVE REGS CX,AX
FA09  50                                PUSH    AX
FA0A  8A E8                             MOV     CH,AL           ; AL=BYTE TO WRITE.
                                                                ;   (MS BIT WRITTEN FIRST)
FA0C  B1 08                             MOV     CL,8            ; FOR 8 DATA BITS IN BYTE.
                                                                ;   NOTE: TWO EDGES PER BIT
FA0E                            W27:                            ; DISASSEMBLE THE DATA BIT
FA0E  D0 D5                             RCL     CH,1            ; ROTATE MS BIT INTO CARRY
FA10  9C                                PUSHF                   ; SAVE FLAGS.
                                                                ;   NOTE: DATA BIT IS IN CARRY
FA11  E8 FA1F R                         CALL    WRITE_BIT       ; WRITE DATA BIT
FA14  9D                                POPF                    ; RESTORE CARRY FOR CRC CALC
FA15  E8 FA3C R                         CALL    CRC_GEN         ; COMPUTE CRC ON DATA BIT
FA18  FE C9                             DEC     CL              ; LOOP TILL ALL 8 BITS DONE
FA1A  75 F2                             JNZ     W27             ; JUMP IF NOT DONE YET
FA1C  58                                POP     AX              ; RESTORE REGS AX,CX
FA1D  59                                POP     CX
FA1E  C3                                RET                     ; WE ARE FINISHED
FA1F                            WRITE_BYTE      ENDP
                                ;-----------------------------------------
FA1F                            WRITE_BIT       PROC    NEAR
                                ; PURPOSE:
                                ;
                                ;  TO WRITE A DATA BIT TO CASSETTE
                                ;  CARRY FLAG CONTAINS DATA BIT
                                ;  I.E. IF SET   DATA BIT IS A ONE
                                ;       IF CLEAR DATA BIT IS A ZERO
                                ;
                                ;  NOTE: TWO EDGES ARE WRITTEN PER BIT
                                ;        ONE BIT  HAS 500 USEC BETWEEN EDGES
                                ;             FOR A 1000 USEC PERIOD (1 MILLISEC)
                                ;
                                ;        ZERO BIT HAS 250 USEC BETWEEN EDGES
                                ;             FOR A  500 USEC PERIOD (.5 MILLISEC)
                                ; CARRY FLAG IS DATA BIT
                                ;-----------------------------------------
                                                                ;ASSUME IT'S A '1'
FA1F  B8 04A0                           MOV     AX,1184         ; SET AX TO NOMINAL ONE SIZE
FA22  72 03                             JC      W28             ; JUMP IF ONE BIT
FA24  B8 0250                           MOV     AX,592          ; NO, SET TO NOMINAL ZERO SIZE
FA27                            W28:                            ; WRITE-BIT-AX
FA27  50                                PUSH    AX              ;WRITE BIT WITH PERIOD EQ TO VALUE
                                                                ; AX
FA28  E4 62                     W29:    IN      AL,PORT_C       ;INPUT TIMER_0 OUTPUT
FA2A  24 20                             AND     AL,020H
FA2C  74 FA                             JZ      W29             ;LOOP TILL HIGH
FA2E  E4 62                     W30:    IN      AL,PORT_C       ;NOW WAIT TILL TIMER'S OUTPUT IS
                                                                ; LOW
FA30  24 20                             AND     AL,020H
FA32  75 FA                             JNZ     W30
                                                                ;RELOAD TIMER WITH PERIOD
                                                                ;FOR NEXT DATA BIT
FA34  58                                POP     AX              ;RESTORE PERIOD COUNT
FA35                            W31:                            ; SET TIMER
FA35  E6 42                             OUT     042H, AL        ; SET LOW BYTE OF TIMER 2
FA37  8A C4                             MOV     AL, AH
FA39  E6 42                             OUT     042H, AL        ; SET HIGH BYTE OF TIMER 2
FA3B  C3                                RET
FA3C                            WRITE_BIT       ENDP
                                ;-----------------------------------------
FA3C                            CRC_GEN         PROC    NEAR
                                ;  UPDATE CRC REGISTER WITH NEXT DATA BIT
                                ;  CRC IS USED TO DETECT READ ERRORS
                                ;  ASSUMES DATA BIT IS IN CARRY
                                ;  REG AX IS MODIFIED
                                ;  FLAGS ARE MODIFIED
                                ;-----------------------------------------
FA3C  A1 0069 R                         MOV     AX,CRC_REG
                                                                ;THE FOLLOWING INSTUCTIONS
                                                                ;WILL SET THE OVERFLOW FLAG
                                                                ;IF CARRY AND MS BIT OF CRC
                                                                ;ARE UNEQUAL
FA3F  D1 D8                             RCR     AX,1
FA41  D1 D0                             RCL     AX,1
FA43  F8                                CLC                     ;CLEAR CARRY
FA44  71 04                             JNO     W32             ;SKIP IF NO OVERFLOW
                                                                ;IF DATA BIT XORED WITH
                                                                ; CRC REG BIT 15 IS ONE
FA46  35 0810                           XOR     AX,0810H        ;THEN XOR CRC REG WITH
                                                                ; 0810H
FA49  F9                                STC                     ;SET CARRY
FA4A  D1 D0                     W32:    RCL     AX,1            ;ROTATE CARRY (DATA BIT)
                                                                ;INTO CRC REG
FA4C  A3 0069 R                         MOV     CRC_REG,AX      ;UPDATE CRC_REG
FA4F  C3                                RET                     ;FINISHED
FA50                            CRC_GEN         ENDP
```

```
                              ;-----------------------------------------------------------------
FA50                          BEGIN_OP       PROC    NEAR          ; START TAPE AND DELAY
FA50  E8 F881 R                       CALL   MOTOR_ON              ;TURN ON MOTOR
FA53  B3 42                           MOV    BL,42H                ;DELAY FOR TAPE DRIVE
                                                                   ;TO GET UP TO SPEED  (1/2 SEC)
FA55  B9 0700                 W33:    MOV    CX,700H               ;INNER LOOP= APPROX. 10 MILLISEC
FA58  E2 FE                   W34:    LOOP   W34
FA5A  FE CB                           DEC    BL
FA5C  75 F7                           JNZ    W33
FA5E  C3                              RET
FA5F                          BEGIN_OP       ENDP
                              ;------ CARRIAGE RETURN, LINE FEED SUBROUTINE
FA5F                          CRLF    PROC   NEAR
FA5F  33 D2                           XOR    DX,DX                 ; PRINTER 0
FA61  32 E4                           XOR    AH,AH                 ; WILL NOW SEND INITIAL LF,CR TO
                                                                   ; PRINTER
FA63  B0 0D                           MOV    AL,0DH                ; CR
FA65  CD 17                           INT    17H                   ; SEND THE LINE FEED
FA67  32 E4                           XOR    AH,AH                 ; NOW FOR THE CR
FA69  B0 0A                           MOV    AL,0AH                ; LF
FA6B  CD 17                           INT    17H                   ; SEND THE CARRIAGE RETURN
FA6D  C3                              RET
FA6E                          CRLF    ENDP
                              ;-----------------------------------------------------------------
                              ;       CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200
                              ;       GRAPHICS FOR CHARACTERS 00H THRU 7FH
                              ;-----------------------------------------------------------------
FA6E                                  ORG    0FA6EH
FA6E                          CRT_CHAR_GEN   LABEL   BYTE
FA6E  00 00 00 00 00 00               DB     000H,000H,000H,000H,000H,000H,000H,000H ; D_00
      00 00
FA76  7E 81 A5 81 BD 99               DB     07EH,081H,0A5H,081H,0BDH,099H,081H,07EH ; D_01
      81 7E
FA7E  7E FF DB FF C3 E7               DB     07EH,0FFH,0DBH,0FFH,0C3H,0E7H,0FFH,07EH ; D_02
      FF 7E
FA86  6C FE FE FE 7C 38               DB     06CH,0FEH,0FEH,0FEH,07CH,038H,010H,000H ; D_03
      10 00
FA8E  10 38 7C FE 7C 38               DB     010H,038H,07CH,0FEH,07CH,038H,010H,000H ; D_04
      10 00
FA96  38 7C 38 FE FE 7C               DB     038H,07CH,038H,0FEH,0FEH,07CH,038H,07CH ; D_05
      38 7C
FA9E  10 10 38 7C FE 7C               DB     010H,010H,038H,07CH,0FEH,07CH,038H,07CH ; D_06
      38 7C
FAA6  00 00 18 3C 3C 18               DB     000H,000H,018H,03CH,03CH,018H,000H,000H ; D_07
      00 00
FAAE  FF FF E7 C3 C3 E7               DB     0FFH,0FFH,0E7H,0C3H,0C3H,0E7H,0FFH,0FFH ; D_08
      FF FF
FAB6  00 3C 66 42 42 66               DB     000H,03CH,066H,042H,042H,066H,03CH,000H ; D_09
      3C 00
FABE  FF C3 99 BD BD 99               DB     0FFH,0C3H,099H,0BDH,0BDH,099H,0C3H,0FFH ; D_0A
      C3 FF
FAC6  0F 07 0F 7D CC CC               DB     00FH,007H,00FH,07DH,0CCH,0CCH,0CCH,078H ; D_0B
      CC 78
FACE  3C 66 66 66 3C 18               DB     03CH,066H,066H,066H,03CH,018H,07EH,018H ; D_0C
      7E 18
FAD6  3F 33 3F 30 30 70               DB     03FH,033H,03FH,030H,030H,070H,0F0H,0E0H ; D_0D
      F0 E0
FADE  7F 63 7F 63 63 67               DB     07FH,063H,07FH,063H,063H,067H,0E6H,0C0H ; D_0E
      E6 C0
FAE6  99 5A 3C E7 E7 3C               DB     099H,05AH,03CH,0E7H,0E7H,03CH,05AH,099H ; D_0F
      5A 99

FAEE  80 E0 F8 FE F8 E0               DB     080H,0E0H,0F8H,0FEH,0F8H,0E0H,080H,000H ; D_10
      80 00
FAF6  02 0E 3E FE 3E 0E               DB     002H,00EH,03EH,0FEH,03EH,00EH,002H,000H ; D_11
      02 00
FAFE  18 3C 7E 18 18 7E               DB     018H,03CH,07EH,018H,018H,07EH,03CH,018H ; D_12
      3C 18
FB06  66 66 66 66 66 00               DB     066H,066H,066H,066H,066H,000H,066H,000H ; D_13
      66 00
FB0E  7F DB DB 7B 1B 1B               DB     07FH,0DBH,0DBH,07BH,01BH,01BH,01BH,000H ; D_14
      1B 00
FB16  3E 63 38 6C 6C 38               DB     03EH,063H,038H,06CH,06CH,038H,0CCH,078H ; D_15
      CC 78
FB1E  00 00 00 00 7E 7E               DB     000H,000H,000H,000H,07EH,07EH,07EH,000H ; D_16
      7E 00
FB26  18 3C 7E 18 7E 3C               DB     018H,03CH,07EH,018H,07EH,03CH,018H,0FFH ; D_17
      18 FF
FB2E  18 3C 7E 18 18 18               DB     018H,03CH,07EH,018H,018H,018H,018H,000H ; D_18
      18 00
FB36  18 18 18 18 7E 3C               DB     018H,018H,018H,018H,07EH,03CH,018H,000H ; D_19
      18 00
FB3E  00 18 0C FE 0C 18               DB     000H,018H,00CH,0FEH,00CH,018H,000H,000H ; D_1A
      00 00
FB46  00 30 60 FE 60 30               DB     000H,030H,060H,0FEH,060H,030H,000H,000H ; D_1B
      00 00
FB4E  00 00 C0 C0 C0 FE               DB     000H,000H,0C0H,0C0H,0C0H,0FEH,000H,000H ; D_1C
      00 00
FB56  00 24 66 FF 66 24               DB     000H,024H,066H,0FFH,066H,024H,000H,000H ; D_1D
      00 00
FB5E  00 18 3C 7E FF FF               DB     000H,018H,03CH,07EH,0FFH,0FFH,000H,000H ; D_1E
      00 00
FB66  00 FF FF 7E 3C 18               DB     000H,0FFH,0FFH,07EH,03CH,018H,000H,000H ; D_1F
      00 00
```

```
FB6E  00 00 00 00 00 00         DB    000H,000H,000H,000H,000H,000H,000H,000H ; SP D_20
      00 00
FB76  30 78 78 30 30 00         DB    030H,078H,078H,030H,030H,000H,030H,000H ; ! D_21
      30 00
FB7E  6C 6C 6C 00 00 00         DB    06CH,06CH,06CH,000H,000H,000H,000H,000H ; " D_22
      00 00
FB86  6C 6C FE 6C FE 6C         DB    06CH,06CH,0FEH,06CH,0FEH,06CH,06CH,000H ; # D_23
      6C 00
FB8E  30 7C C0 78 0C F8         DB    030H,07CH,0C0H,078H,00CH,0F8H,030H,000H ; $ D_24
      30 00
FB96  00 C6 CC 18 30 66         DB    000H,0C6H,0CCH,018H,030H,066H,0C6H,000H ;
      C6 00
                                                                  ; PER CENT D_25
FB9E  38 6C 38 76 DC CC         DB    038H,06CH,038H,076H,0DCH,0CCH,076H,000H ; & D_26
      76 00
FBA6  60 60 C0 00 00 00         DB    060H,060H,0C0H,000H,000H,000H,000H,000H ; ' D_27
      00 00
FBAE  18 30 60 60 60 30         DB    018H,030H,060H,060H,060H,030H,018H,000H ; ( D_28
      18 00
FBB6  60 30 18 18 18 30         DB    060H,030H,018H,018H,018H,030H,060H,000H ; ) D_29
      60 00
FBBE  00 66 3C FF 3C 66         DB    000H,066H,03CH,0FFH,03CH,066H,000H,000H ; * D_2A
      00 00
FBC6  00 30 30 FC 30 30         DB    000H,030H,030H,0FCH,030H,030H,000H,000H ; + D_2B
      00 00
FBCE  00 00 00 00 00 30         DB    000H,000H,000H,000H,000H,030H,030H,060H ; , D_2C
      30 60
FBD6  00 00 00 FC 00 00         DB    000H,000H,000H,0FCH,000H,000H,000H,000H ;- D_2D
      00 00
FBDE  00 00 00 00 00 30         DB    000H,000H,000H,000H,000H,030H,030H,000H ; . D_2E
      30 00
FBE6  06 0C 18 30 60 C0         DB    006H,00CH,018H,030H,060H,0C0H,080H,000H ; / D_2F
      80 00

FBEE  7C C6 CE DE F6 E6         DB    07CH,0C6H,0CEH,0DEH,0F6H,0E6H,07CH,000H ; 0 D_30
      7C 00
FBF6  30 70 30 30 30 30         DB    030H,070H,030H,030H,030H,030H,0FCH,000H ; 1 D_31
      FC 00
FBFE  78 CC 0C 38 60 CC         DB    078H,0CCH,00CH,038H,060H,0CCH,0FCH,000H ; 2 D_32
      FC 00
FC06  78 CC 0C 38 0C CC         DB    078H,0CCH,00CH,038H,00CH,0CCH,078H,000H ; 3 D_33
      78 00
FC0E  1C 3C 6C CC FE 0C         DB    01CH,03CH,06CH,0CCH,0FEH,00CH,01EH,000H ; 4 D_34
      1E 00
FC16  FC C0 F8 0C 0C CC         DB    0FCH,0C0H,0F8H,00CH,00CH,0CCH,078H,000H ; 5 D_35
      78 00
FC1E  38 60 C0 F8 CC CC         DB    038H,060H,0C0H,0F8H,0CCH,0CCH,078H,000H ; 6 D_36
      78 00
FC26  FC CC 0C 18 30 30         DB    0FCH,0CCH,00CH,018H,030H,030H,030H,000H ; 7 D_37
      30 00
FC2E  78 CC CC 78 CC CC         DB    078H,0CCH,0CCH,078H,0CCH,0CCH,078H,000H ; 8 D_38
      78 00
FC36  78 CC CC 7C 0C 18         DB    078H,0CCH,0CCH,07CH,00CH,018H,070H,000H ; 9 D_39
      70 00
FC3E  00 30 30 00 00 30         DB    000H,030H,030H,000H,000H,030H,030H,000H ; : D_3A
      30 00
FC46  00 30 30 00 00 30         DB    000H,030H,030H,000H,000H,030H,030H,060H ; ; D_3B
      30 60
FC4E  18 30 60 C0 60 30         DB    018H,030H,060H,0C0H,060H,030H,018H,000H ; < D_3C
      18 00
FC56  00 00 FC 00 00 FC         DB    000H,000H,0FCH,000H,000H,0FCH,000H,000H ; = D_3D
      00 00
FC5E  60 30 18 0C 18 30         DB    060H,030H,018H,00CH,018H,030H,060H,000H ; > D_3E
      60 00
FC66  78 CC 0C 18 30 00         DB    078H,0CCH,00CH,018H,030H,000H,030H,000H ; ? D_3F
      30 00

FC6E  7C C6 DE DE DE C0         DB    07CH,0C6H,0DEH,0DEH,0DEH,0C0H,078H,000H ; @ D_40
      78 00
FC76  30 78 CC CC FC CC         DB    030H,078H,0CCH,0CCH,0FCH,0CCH,0CCH,000H ; A D_41
      CC 00
FC7E  FC 66 66 7C 66 66         DB    0FCH,066H,066H,07CH,066H,066H,0FCH,000H ; B D_42
      FC 00
FC86  3C 66 C0 C0 C0 66         DB    03CH,066H,0C0H,0C0H,0C0H,066H,03CH,000H ; C D_43
      3C 00
FC8E  F8 6C 66 66 66 6C         DB    0F8H,06CH,066H,066H,066H,06CH,0F8H,000H ; D D_44
      F8 00
FC96  FE 62 68 78 68 62         DB    0FEH,062H,068H,078H,068H,062H,0FEH,000H ; E D_45
      FE 00
FC9E  FE 62 68 78 68 60         DB    0FEH,062H,068H,078H,068H,060H,0F0H,000H ; F D_46
      F0 00
FCA6  3C 66 C0 C0 CE 66         DB    03CH,066H,0C0H,0C0H,0CEH,066H,03EH,000H ; G D_47
      3E 00
FCAE  CC CC CC FC CC CC         DB    0CCH,0CCH,0CCH,0FCH,0CCH,0CCH,0CCH,000H ; H D_48
      CC 00
FCB6  78 30 30 30 30 30         DB    078H,030H,030H,030H,030H,030H,078H,000H ; I D_49
      78 00
FCBE  1E 0C 0C 0C CC CC         DB    01EH,00CH,00CH,00CH,0CCH,0CCH,078H,000H ; J D_4A
      78 00
FCC6  E6 66 6C 78 6C 66         DB    0E6H,066H,06CH,078H,06CH,066H,0E6H,000H ; K D_4B
      E6 00
FCCE  F0 60 60 60 62 66         DB    0F0H,060H,060H,060H,062H,066H,0FEH,000H ; L D_4C
      FE 00
FCD6  C6 EE FE FE D6 C6         DB    0C6H,0EEH,0FEH,0FEH,0D6H,0C6H,0C6H,000H ; M D_4D
      C6 00
FCDE  C6 E6 F6 DE CE C6         DB    0C6H,0E6H,0F6H,0DEH,0CEH,0C6H,0C6H,000H ; N D_4E
      C6 00
FCE6  38 6C C6 C6 C6 6C         DB    038H,06CH,0C6H,0C6H,0C6H,06CH,038H,000H ; O D_4F
      38 00
```

```
FCEE  FC 66 66 7C 60 60          DB      0FCH,066H,066H,07CH,060H,060H,0F0H,000H ; P D_50
      F0 00
FCF6  78 CC CC CC DC 78          DB      078H,0CCH,0CCH,0CCH,0DCH,078H,01CH,000H ; Q D_51
      1C 00
FCFE  FC 66 66 7C 6C 66          DB      0FCH,066H,066H,07CH,06CH,066H,0E6H,000H ; R D_52
      E6 00
FD06  78 CC E0 70 1C CC          DB      078H,0CCH,0E0H,070H,01CH,0CCH,078H,000H ; S D_53
      78 00
FD0E  FC B4 30 30 30 30          DB      0FCH,0B4H,030H,030H,030H,030H,078H,000H ; T D_54
      78 00
FD16  CC CC CC CC CC CC          DB      0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,0FCH,000H ; U D_55
      FC 00
FD1E  CC CC CC CC CC 78          DB      0CCH,0CCH,0CCH,0CCH,0CCH,078H,030H,000H ; V D_56
      30 00
FD26  C6 C6 C6 D6 FE EE          DB      0C6H,0C6H,0C6H,0D6H,0FEH,0EEH,0C6H,000H ; W D_57
      C6 00
FD2E  C6 C6 6C 38 38 6C          DB      0C6H,0C6H,06CH,038H,038H,06CH,0C6H,000H ; X D_58
      C6 00
FD36  CC CC CC 78 30 30          DB      0CCH,0CCH,0CCH,078H,030H,030H,078H,000H ; Y D_59
      78 00
FD3E  FE C6 8C 18 32 66          DB      0FEH,0C6H,08CH,018H,032H,066H,0FEH,000H ; Z D_5A
      FE 00
FD46  78 60 60 60 60 60          DB      078H,060H,060H,060H,060H,060H,078H,000H ; [ D_5B
      78 00
FD4E  C0 60 30 18 0C 06          DB      0C0H,060H,030H,018H,00CH,006H,002H,000H ;
      02 00
                                                                 ; BACKSLASH D_5C
FD56  78 18 18 18 18 18          DB      078H,018H,018H,018H,018H,018H,078H,000H ; ] D_5D
      78 00
FD5E  10 38 6C C6 00 00          DB      010H,038H,06CH,0C6H,000H,000H,000H,000H ;
      00 00
                                                                 ; CIRCUMFLEX D_5E
FD66  00 00 00 00 00 00          DB      000H,000H,000H,000H,000H,000H,000H,0FFH ; _ D_5F
      00 FF

FD6E  30 30 18 00 00 00          DB      030H,030H,018H,000H,000H,000H,000H,000H ; ` D_60
      00 00
FD76  00 00 78 0C 7C CC          DB      000H,000H,078H,00CH,07CH,0CCH,076H,000H ;
      76 00
                                                                 ; LOWER CASE A D_61
FD7E  E0 60 60 7C 66 66          DB      0E0H,060H,060H,07CH,066H,066H,0DCH,000H ; LC B D_62
      DC 00
FD86  00 00 78 CC C0 CC          DB      000H,000H,078H,0CCH,0C0H,0CCH,078H,000H ; LC C D_63
      78 00
FD8E  1C 0C 0C 7C CC CC          DB      01CH,00CH,00CH,07CH,0CCH,0CCH,076H,000H ; LC D D_64
      76 00
FD96  00 00 78 CC FC C0          DB      000H,000H,078H,0CCH,0FCH,0C0H,078H,000H ; LC E D_65
      78 00
FD9E  38 6C 60 F0 60 60          DB      038H,06CH,060H,0F0H,060H,060H,0F0H,000H ; LC F D_66
      F0 00
FDA6  00 00 76 CC CC 7C          DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,0F8H ; LC G D_67
      0C F8
FDAE  E0 60 6C 76 66 66          DB      0E0H,060H,06CH,076H,066H,066H,0E6H,000H ; LC H D_68
      E6 00
FDB6  30 00 70 30 30 30          DB      030H,000H,070H,030H,030H,030H,078H,000H ; LC I D_69
      78 00
FDBE  0C 00 0C 0C 0C CC          DB      00CH,000H,00CH,00CH,00CH,0CCH,0CCH,078H ; LC J D_6A
      CC 78
FDC6  E0 60 66 6C 78 6C          DB      0E0H,060H,066H,06CH,078H,06CH,0E6H,000H ; LC K D_6B
      E6 00
FDCE  70 30 30 30 30 30          DB      070H,030H,030H,030H,030H,030H,078H,000H ; LC L D_6C
      78 00
FDD6  00 00 CC FE FE D6          DB      000H,000H,0CCH,0FEH,0FEH,0D6H,0C6H,000H ; LC M D_6D
      C6 00
FDDE  00 00 F8 CC CC CC          DB      000H,000H,0F8H,0CCH,0CCH,0CCH,0CCH,000H ; LC N D_6E
      CC 00
FDE6  00 00 78 CC CC CC          DB      000H,000H,078H,0CCH,0CCH,0CCH,078H,000H ; LC O D_6F
      78 00

FDEE  00 00 DC 66 66 7C          DB      000H,000H,0DCH,066H,066H,07CH,060H,0F0H ; LC P D_70
      60 F0
FDF6  00 00 76 CC CC 7C          DB      000H,000H,076H,0CCH,0CCH,07CH,00CH,01EH ; LC Q D_71
      0C 1E
FDFE  00 00 DC 76 66 60          DB      000H,000H,0DCH,076H,066H,060H,0F0H,000H ; LC R D_72
      F0 00
FE06  00 00 7C C0 78 0C          DB      000H,000H,07CH,0C0H,078H,00CH,0F8H,000H ; LC S D_73
      F8 00
FE0E  10 30 7C 30 30 34          DB      010H,030H,07CH,030H,030H,034H,018H,000H ; LC T D_74
      18 00
FE16  00 00 CC CC CC CC          DB      000H,000H,0CCH,0CCH,0CCH,0CCH,076H,000H ; LC U D_75
      76 00
FE1E  00 00 CC CC CC 78          DB      000H,000H,0CCH,0CCH,0CCH,078H,030H,000H ; LC V D_76
      30 00
FE26  00 00 C6 D6 FE FE          DB      000H,000H,0C6H,0D6H,0FEH,0FEH,06CH,000H ; LC W D_77
      6C 00
FE2E  00 00 C6 6C 38 6C          DB      000H,000H,0C6H,06CH,038H,06CH,0C6H,000H ; LC X D_78
      C6 00
FE36  00 00 CC CC CC 7C          DB      000H,000H,0CCH,0CCH,0CCH,07CH,00CH,0F8H ; LC Y D_79
      0C F8
FE3E  00 00 FC 98 30 64          DB      000H,000H,0FCH,098H,030H,064H,0FCH,000H ; LC Z D_7A
      FC 00
FE46  1C 30 30 E0 30 30          DB      01CH,030H,030H,0E0H,030H,030H,01CH,000H ; { D_7B
      1C 00
FE4E  18 18 18 00 18 18          DB      018H,018H,018H,000H,018H,018H,018H,000H ; : D_7C
      18 00
FE56  E0 30 30 1C 30 30          DB      0E0H,030H,030H,01CH,030H,030H,0E0H,000H ; } D_7D
      E0 00
FE5E  76 DC 00 00 00 00          DB      076H,0DCH,000H,000H,000H,000H,000H,000H ; ~ D_7E
      00 00
FE66  00 10 38 6C C6 C6          DB      000H,010H,038H,06CH,0C6H,0C6H,0FEH,000H ;
      FE 00

                                                                 ; DELTA D_7F
```

Appendix A

```
FE6E                                    ORG     0FE6EH
FE6E  E9 1393 R                         JMP     NEAR PTR TIME_OF_DAY
                                ;------------------------------------------------------------
                                ;               CRC CHECK/GENERATION ROUTINE
                                ;       ROUTINE TO CHECK A ROM MODULE USING THE POLYNOMIMAL:
                                ;                  X16 + X12 + X5 + 1
                                ;       CALLING PARAMETERS:
                                ;       DS = DATA SEGMENT OF ROM SPACE TO BE CHECKED
                                ;       SI = INDEX OFFSET INTO DS POINTING TO 1ST BYTE
                                ;       CX = LENGTH OF SPACE TO BE CHECKED (INCLUDING CRC BYTES)
                                ;       ON EXIT:
                                ;       ZERO FLAG = SET = CRC CHECKED OK
                                ;       AH = 00
                                ;       AL = ??
                                ;       BX = 0000
                                ;       CL = 04
                                ;       DX = 0000 IF CRC CHECKED OK, ELSE, ACCUMULATED CRC
                                ;       SI = (SI(ENTRY)+BX(ENTRY)
                                ;       NOTE: ROUTINE WILL RETURN IMMEDIATLY IF "RESET_FLAG
                                ;             IS EQUAL TO "1234H" (WARM START)
                                ;------------------------------------------------------------
                                ;
FE71                            CRC_CHECK       PROC    NEAR
                                        ASSUME  DS:NOTHING
FE71  8B D9                             MOV     BX,CX           ; SAVE COUNT
FE73  BA FFFF                           MOV     DX,0FFFFH       ; INIT. ENCODE REGISTER
FE76  FC                                CLD                     ; SET DIR FLAG TO INCREMENT
FE77  32 E4                             XOR     AH,AH           ; INIT. WORK REG HIGH
FE79  B1 04                             MOV     CL,4            ; SET ROTATE COUNT
FE7B  AC                        CRC_1:  LODSB                   ; GET A BYTE
FE7C  32 F0                             XOR     DH,AL           ; FORM Aj + Cj + 1
FE7E  8A C6                             MOV     AL,DH
FE80  D3 C0                             ROL     AX,CL           ; SHIFT WORK REG BACK 4
FE82  33 D0                             XOR     DX,AX           ; ADD INTO RESULT REG
FE84  D1 C0                             ROL     AX,1            ; SHIFT WORK REG BACK 1
FE86  86 F2                             XCHG    DH,DL           ; SWAP PARTIAL SUM INTO RESULT REG
FE88  33 D0                             XOR     DX,AX           ; ADD WORK REG INTO RESULTS
FE8A  D3 C8                             ROR     AX,CL           ; SHIFT WORK REG OVER 4
FE8C  24 E0                             AND     AL,11100000B    ; CLEAR OFF (EFGH)
FE8E  33 D0                             XOR     DX,AX           ; ADD (ABCD) INTO RESULTS
FE90  D1 C8                             ROR     AX,1            ; SHIFT WORK REG ON OVER (AH=0 FOR
                                                                ; NEXT PASS)
FE92  32 F0                             XOR     DH,AL           ; ADD (ABCD INTO RESULTS LOW)
FE94  4B                                DEC     BX              ; DECREMENT COUNT
FE95  75 E4                             JNZ     CRC_1           ; LOOP TILL COUNT = 0000
FE97  0B D2                             OR      DX,DX           ; DX S/B = 0000 IF O.K.
FE99  C3                                RET                     ; RETURN TO CALLER
FE9A                            CRC_CHECK       ENDP
                                ;--------------------------------------------------------------
                                ;   SUBROUTINE TO READ AN 8250 REGISTER.  MAY ALSO BUMP ERROR
                                ;       REPORTER (BL) AND/OR REG DX (PORT ADDRESS) DEPENDING ON
                                ;       WHICH ENTRY POINT IS CHOSEN.
                                ;   THIS SUBROUTINE WAS WRITTEN TO AVOID MULTIPLE USE OF I/O TIME
                                ;       DELAYS FOR THE 8250.  IT WAS THE MOST EFFICIENT WAY TO
                                ;       INCLUDE THE DELAYS.
                                ;   IN EVERY CASE, UPON RETURN, REG AL WILL CONTAIN THE CONTENTS OF
                                ;       PORT(DX)
                                ;--------------------------------------------------------------
FE9A                            RR1     PROC    NEAR
FE9A  32 C0                             XOR     AL,AL
FE9C  EE                                OUT     DX,AL           ; DISABLE ALL INTERRUPTS
FE9D  FE C3                             INC     BL              ; BUMP ERROR REPORTER
FE9F  42                        RR2:    INC     DX              ; INCR PORT ADDR
FEA0  EC                        RR3:    IN      AL,DX           ; READ REGISTER
FEA1  C3                                RET
FEA2                            RR1     ENDP
                                ;-----------------------------------------------
                                ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM
                                ; CHANNEL 0 OF THE 8253 TIMER.  INPUT FREQUENCY IS 1.19318 MHZ
                                ; AND THE DIVISOR IS 65536, RESULTING IN APPROX. 18.2 INTERRUPTS
                                ; EVERY SECOND.
                                ;
                                ; THE INTERRUPT HANDLER MAINTAINS A COUNT OF INTERRUPTS SINCE POWER
                                ;  ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY.
                                ;  INTERRUPTS MISSED WHILE INTS. WERE DISABLED ARE TAKEN CARE OF
                                ;  BY THE USE OF TIMER 1 AS A OVERFLOW COUNTER
                                ; THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT
                                ;  OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE DISKETTE
                                ;  MOTOR, AND RESET THE MOTOR RUNNING FLAGS
                                ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH
                                ;  INTERRUPT 1CH AT EVERY TIME TICK.  THE USER MUST CODE A ROUTINE
                                ; AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE.
                                ;-----------------------------------------------
FEA5                                    ORG     0FEA5H
                                        ASSUME  DS:DATA
FEA5                            TIMER_INT       PROC    FAR
FEA5  FB                                STI                     ; INTERRUPTS BACK ON
FEA6  1E                                PUSH    DS
FEA7  50                                PUSH    AX
FEA8  52                                PUSH    DX              ; SAVE MACHINE STATE
FEA9  E8 138B R                         CALL    DDS
FEAC  FF 06 006C R                      INC     TIMER_LOW       ; INCREMENT TIME
FEB0  75 04                             JNZ     T4              ; TEST_DAY
FEB2  FF 06 006E R                      INC     TIMER_HIGH      ; INCREMENT HIGH WORD OF TIME
FEB6                            T4:                             ; TEST_DAY
FEB6  83 3E 006E R 18                   CMP     TIMER_HIGH,018H ; TEST FOR COUNT EQUALLING 24 HOURS
FEBB  75 15                             JNZ     T5              ; DISKETTE_CTL
FEBD  81 3E 006C R 00B0                 CMP     TIMER_LOW,0B0H
FEC3  75 0D                             JNZ     T5              ; DISKETTE_CTL
```

```
                                ;------ TIMER HAS GONE 24 HOURS
FEC5  2B C0                             SUB     AX,AX
FEC7  A3 006E R                         MOV     TIMER_HIGH,AX
FECA  A3 006C R                         MOV     TIMER_LOW,AX
FECD  C6 06 0070 R 01                   MOV     TIMER_OFL,1
                                ;------ TEST FOR DISKETTE TIME OUT
FED2                            T5:                             ; LOOP TILL ALL OVERFLOWS TAKEN
                                                                ; CARE OF
FED2  FE 0E 0040 R                      DEC     MOTOR_COUNT
FED6  75 09                             JNZ     T6              ; RETURN IF COUNT NOT OUT
FED8  80 26 003F R F0                   AND     MOTOR_STATUS,0F0H ; TURN OFF MOTOR RUNNING BITS
FEDD  B0 80                             MOV     AL,FDC_RESET    ; TURN OFF MOTOR, DO NOT RESET FDC
FEDF  E6 F2                             OUT     NEC_CTL,AL      ; TURN OFF THE MOTOR
FEE1  CD 1C                     T6:     INT     1CH             ; TRANSFER CONTROL TO A USER
                                                                ; ROUTINE
FEE3  B0 20                             MOV     AL,EOI
FEE5  E6 20                             OUT     020H,AL         ; END OF INTERRUPT TO 8259
FEE7  5A                                POP     DX
FEE8  58                                POP     AX
FEE9  1F                                POP     DS              ; RESET MACHINE STATE
FEEA  CF                                IRET                    ; RETURN FROM INTERRUPT
FEEB                            TIMER_INT       ENDP
                                ;--------------------------------------------------------------
                                ;    ARITHMETIC CHECKSUM  ROUTINE
                                ;       ENTRY:
                                ;       DS = DATA SEGMENT OF ROM SPACE TO BE CHECKED
                                ;       SI = INDEX OFFSET INTO DS POINTING TO 1ST BYTE
                                ;       CX = LENGTH OF SPACE TO BE CHECKED
                                ;       EXIT: ZERO FLAG OFF=ERROR, ON= SPACE CHECKED OK
                                ;--------------------------------------------------------------
FEEB                            ROS_CHECKSUM    PROC    NEAR
FEEB  02 04                     RC_0:   ADD     AL,DS:[SI]
FEED  46                                INC     SI
FEEE  E2 FB                             LOOP    RC_0
FEF0  0A C0                             OR      AL,AL
FEF2  C3                                RET
FEF3                            ROS_CHECKSUM    ENDP
                                ;--------------------------------------------
                                ; THESE ARE THE VECTORS WHICH ARE MOVED INTO    :
                                ; THE 8086 INTERRUPT AREA DURING POWER ON.      :
                                ; ONLY THE OFFSETS ARE DISPLAYED HERE, CODE     :
                                ; SEGMENT WILL BE ADDED FOR ALL OF THEM, EXCEPT :
                                , WHERE NOTED.                                  :
                                ;--------------------------------------------
                                        ASSUME  CS:CODE
FEF3                                    ORG     0FEF3H
FEF3                            VECTOR_TABLE    LABEL   WORD ; VECTOR TABLE FOR MOVE TO INTERRUPTS
FEF3  FEA5 R                            DW      OFFSET TIMER_INT ; INTERRUPT 8
FEF5  1561 R                            DW      OFFSET  KB_INT  ; INTERRUPT 9
FEF7  F815 R                            DW      OFFSET  D11     ; INTERRUPT A
FEF9  F815 R                            DW      OFFSET  D11     ; INTERRUPT B
FEFB  F815 R                            DW      OFFSET  D11     ; INTERRUPT C
FEFD  F815 R                            DW      OFFSET  D11     ; INTERRUPT D
FEFF  EF57 R                            DW      OFFSET DISK_INT ; INTERRUPT E
FF01  F815 R                            DW      OFFSET  D11     ; INTERRUPT F
FF03  0D0B R                            DW      OFFSET VIDEO_IO ; INTERRUPT 10H
FF05  F84D R                            DW      OFFSET EQUIPMENT ; INTERRUPT 11H
FF07  F841 R                            DW      OFFSET MEMORY_SIZE_DETERMINE ; INTERRUPT 12H
FF09  EC59 R                            DW      OFFSET DISKETTE_IO ; INTERRUPT 13H
FF0B  E739 R                            DW      OFFSET RS232_IO ; INTERRUPT 14H
FF0D  F859 R                            DW      CASSETTE_IO     ; INTERRUPT 15H
FF0F  13DD R                            DW      OFFSET KEYBOARD_IO ; INTERRUPT 16H
FF11  EFD2 R                            DW      OFFSET PRINTER_IO ; INTERRUPT 17H
FF13  0000                              DW      00000H          ; INTERRUPT 18H
                                        DW      0F600H          ; MUST BE INSERTED INTO TABLE LATER
FF15  0B1B R                            DW      OFFSET BOOT_STRAP ; INTERRUPT 19H
FF17  1393 R                            DW      TIME_OF_DAY     ; INTERRUPT 1AH -- TIME OF DAY
FF19  F83C R                            DW      DUMMY_RETURN    ; INTERRUPT 1BH -- KEYBD BREAK ADDR
FF1B  F83C R                            DW      DUMMY_RETURN    ; INTERRUPT 1C -- TIMER BREAK ADDR
FF1D  F0A4 R                            DW      VIDEO_PARMS     ; INTERRUPT 1D -- VIDEO PARAMETERS
FF1F  EFC7 R                            DW      OFFSET DISK_BASE ; INTERRUPT 1E -- DISK PARMS
FF21  E05E R                            DW      CRT_CHARH       ; INTERRUPT 1F --  VIDEO EXT
FF23                            P_MSG   PROC    NEAR
FF23  2E: 8A 04                 G12A:   MOV     AL,CS:[SI]      ; PUT CHAR IN AL
FF26  46                                INC     SI              ; POINT TO NEXT CHAR
FF27  50                                PUSH    AX              ; SAVE PRINT CHAR
FF28  E8 188A R                         CALL    PRT_HEX         ; CALL VIDEO_IO
FF2B  58                                POP     AX              ; RECOVER PRINT CHAR
FF2C  3C 0D                             CMP     AL,13           ; WAS IT CARRAGE RETURN?
FF2E  75 F3                             JNE     G12A            ; NO,KEEP PRINTING STRING
FF30  C3                                RET
FF31                            P_MSG   ENDP
                                ;       ROUTINE TO SOUND BEEPER
FF31                            BEEP    PROC    NEAR
FF31  B0 B6                             MOV     AL,10110110B    ; SEL TIM 2,LSB,MSB,BINARY
FF33  E6 43                             OUT     TIMER+3,AL      ; WRITE THE TIMER MODE REG
FF35  B8 0533                           MOV     AX,533H         ; DIVISOR FOR 1000 HZ
FF38  E6 42                             OUT     TIMER+2,AL      ; WRITE TIMER 2 CNT - LSB
FF3A  8A C4                             MOV     AL,AH
FF3C  E6 42                             OUT     TIMER+2,AL      ; WRITE TIMER 2 CNT - MSB
FF3E  E4 61                             IN      AL,PORT_B       ; GET CURRENT SETTING OF PORT
FF40  8A E0                             MOV     AH,AL           ; SAVE THAT SETTING
FF42  0C 03                             OR      AL,03           ; TURN SPEAKER ON
FF44  E6 61                             OUT     PORT_B,AL
FF46  2B C9                             SUB     CX,CX           ; SET CNT TO WAIT 500 MS
FF48  E2 FE                     G7:     LOOP    G7              ; DELAY BEFORE TURNING OFF
FF4A  FE CB                             DEC     BL              ; DELAY CNT EXPIRED?
FF4C  75 FA                             JNZ     G7              ; NO - CONTINUE BEEPING SPK
FF4E  8A C4                             MOV     AL,AH           ; RECOVER VALUE OF PORT
FF50  E6 61                             OUT     PORT_B,AL
FF52  C3                                RET                     ; RETURN TO CALLER
FF53                            BEEP    ENDP
```

```
                          ;-----------------------------------
                          ; DUMMY RETURN FOR ADDRESS COMPATIBILITY
                          ;-----------------------------------
FF53                              ORG     0FF53H
FF53  CF                          IRET
                          ;-- INT 5 ----------------------------------------------------
                          ;       THIS LOGIC WILL BE INVOKED BY INTERRUPT 05H TO PRINT
                          ;       THE SCREEN. THE CURSOR POSITION AT THE TIME THIS ROUTINE
                          ;       IS INVOKED WILL BE SAVED AND RESTORED UPON COMPLETION. THE
                          ;       ROUTINE IS INTENDED TO RUN WITH INTERRUPTS ENABLED.
                          ;       IF A SUBSEQUENT 'PRINT SCREEN KEY IS DEPRESSED DURING THE
                          ;       TIME THIS ROUTINE IS PRINTING IT WILL BE IGNORED.
                          ;       ADDRESS 50:0 CONTAINS THE STATUS OF THE PRINT SCREEN:
                          ;
                          ;       50:0    =0      EITHER PRINT SCREEN HAS NOT BEEN CALLED
                          ;                       OR UPON RETURN FROM A CALL THIS INDICATES
                          ;                       A SUCCESSFUL OPERATION.
                          ;
                          ;               =1      PRINT SCREEN IS IN PROGRESS
                          ;
                          ;               =0FFH   ERROR ENCOUNTERED DURING PRINTING
                          ;-------------------------------------------------------------
                                  ASSUME  CS:CODE,DS:XXDATA
FF54                              ORG     0FF54H
FF54                      PRINT_SCREEN    PROC    FAR
FF54  FB                          STI                       ; MUST RUN WITH INTERRUPTS ENABLED
FF55  1E                          PUSH    DS                ; MUST USE 50:0 FOR DATA AREA
                                                            ; STORAGE
FF56  50                          PUSH    AX
FF57  53                          PUSH    BX
FF58  51                          PUSH    CX                ; WILL USE THIS LATER FOR CURSOR
                                                            ; LIMITS
FF59  52                          PUSH    DX                ; WILL HOLD CURRENT CURSOR POSITION
FF5A  B8 ---- R                   MOV     AX,XXDATA         ; HEX 50
FF5D  8E D8                       MOV     DS,AX
FF5F  80 3E 0000 R 01             CMP     STATUS_BYTE,1     ; SEE IF PRINT ALREADY IN PROGRESS
FF64  74 5F                       JZ      EXIT              ; JUMP IF PRINT ALREADY IN PROGRESS
FF66  C6 06 0000 R 01             MOV     STATUS_BYTE,1     ; INDICATE PRINT NOW IN PROGRESS
FF6B  B4 0F                       MOV     AH,15             ; WILL REQUEST THE CURRENT SCREEN
                                                            ; MODE
FF6D  CD 10                       INT     10H               ;       [AL]=MODE
                                                            ;       [AH]=NUMBER COLUMNS/LINE
                                                            ;       [BH]=VISUAL PAGE
                          ; ***********************************************************
                          ;       AT THIS POINT WE KNOW THE COLUMNS/LINE ARE IN
                          ;       [AX] AND THE PAGE IF APPLICABLE IS IN [BH]. THE STACK
                          ;       HAS DS,AX,BX,CX,DX PUSHED. [AL] HAS VIDEO MODE
                          ; ***********************************************************
FF6F  8A CC                       MOV     CL,AH             ; WILL MAKE USE OF [CX] REGISTER TO
FF71  B5 19                       MOV     CH,25             ; CONTROL ROW & COLUMNS
FF73  E8 FA5F R                   CALL    CRLF              ; CARRIAGE RETURN LINE FEED ROUTINE
FF76  51                          PUSH    CX                ; SAVE SCREEN BOUNDS
FF77  B4 03                       MOV     AH,3              ; WILL NOW READ THE CURSOR.
FF79  CD 10                       INT     10H               ; AND PRESERVE THE POSITION
FF7B  59                          POP     CX                ; RECALL SCREEN BOUNDS
FF7C  52                          PUSH    DX                ; RECALL [BH]=VISUAL PAGE
FF7D  33 D2                       XOR     DX,DX             ; WILL SET CURSOR POSITION TO [0,0]
                          ; ****************************************************************
                          ;       THE LOOP FROM PRI10 TO THE INSTRUCTION PRIOR TO PRI20
                          ;       IS THE LOOP TO READ EACH CURSOR POSITION FROM THE SCREEN
                          ;       AND PRINT.
                          ; ****************************************************************
FF7F  B4 02               PRI10:  MOV     AH,2              ; TO INDICATE CURSOR SET REQUEST
FF81  CD 10                       INT     10H               ; NEW CURSOR POSITION ESTABLISHED
FF83  B4 08                       MOV     AH,8              ; TO INDICATE READ CHARACTER
FF85  CD 10                       INT     10H               ; CHARACTER NOW IN [AL]
FF87  0A C0                       OR      AL,AL             ; SEE IF VALID CHAR
FF89  75 02                       JNZ     PRI15             ; JUMP IF VALID CHAR
FF8B  B0 20                       MOV     AL,' '            ; MAKE A BLANK
FF8D  52                  PRI15:  PUSH    DX                ; SAVE CURSOR POSITION
FF8E  33 D2                       XOR     DX,DX             ; INDICATE PRINTER 1
FF90  32 E4                       XOR     AH,AH             ; TO INDICATE PRINT CHAR IN [AL]
FF92  CD 17                       INT     17H               ; PRINT THE CHARACTER
FF94  5A                          POP     DX                ; RECALL CURSOR POSITION
FF95  F6 C4 29                    TEST    AH,029H           ; TEST FOR PRINTER ERROR
FF98  75 21                       JNZ     ERR10             ; JUMP IF ERROR DETECTED
FF9A  FE C2                       INC     DL                ; ADVANCE TO NEXT COLUMN
FF9C  3A CA                       CMP     CL,DL             ; SEE IF AT END OF LINE
FF9E  75 DF                       JNZ     PRI10             ; IF NOT PROCEED
FFA0  32 D2                       XOR     DL,DL             ; BACK TO COLUMN 0
FFA2  8A E2                       MOV     AH,DL             ; [AH]=0
FFA4  52                          PUSH    DX                ; SAVE NEW CURSOR POSITION
FFA5  E8 FA5F R                   CALL    CRLF              ; LINE FEED CARRIAGE RETURN
FFA8  5A                          POP     DX                ; RECALL CURSOR POSITION
FFA9  FE C6                       INC     DH                ; ADVANCE TO NEXT LINE
FFAB  3A EE                       CMP     CH,DH             ; FINISHED?
FFAD  75 D0                       JNZ     PRI10             ; IF NOT CONTINUE
FFAF  5A                          POP     DX                ; RECALL CURSOR POSITION
FFB0  B4 02                       MOV     AH,2              ; TO INDICATE CURSOR SET REQUEST
FFB2  CD 10                       INT     10H               ; CURSOR POSITION RESTORED
FFB4  C6 06 0000 R 00             MOV     STATUS_BYTE,0     ; INDICATE FINISHED
FFB9  EB 0A                       JMP     SHORT EXIT        ; EXIT THE ROUTINE
FFBB  5A                  ERR10:  POP     DX                ; GET CURSOR POSITION
FFBC  B4 02                       MOV     AH,2              ; TO REQUEST CURSOR SET
FFBE  CD 10                       INT     10H               ; CURSOR POSITION RESTORED
FFC0  C6 06 0000 R FF             MOV     STATUS_BYTE,0FFH          ; INDICATE ERROR
FFC5  5A                  EXIT:   POP     DX                ; RESTORE ALL THE REGISTERS USED
FFC6  59                          POP     CX
FFC7  5B                          POP     BX
FFC8  58                          POP     AX
FFC9  1F                          POP     DS
FFCA  CF                          IRET
FFCB                      PRINT_SCREEN    ENDP
```

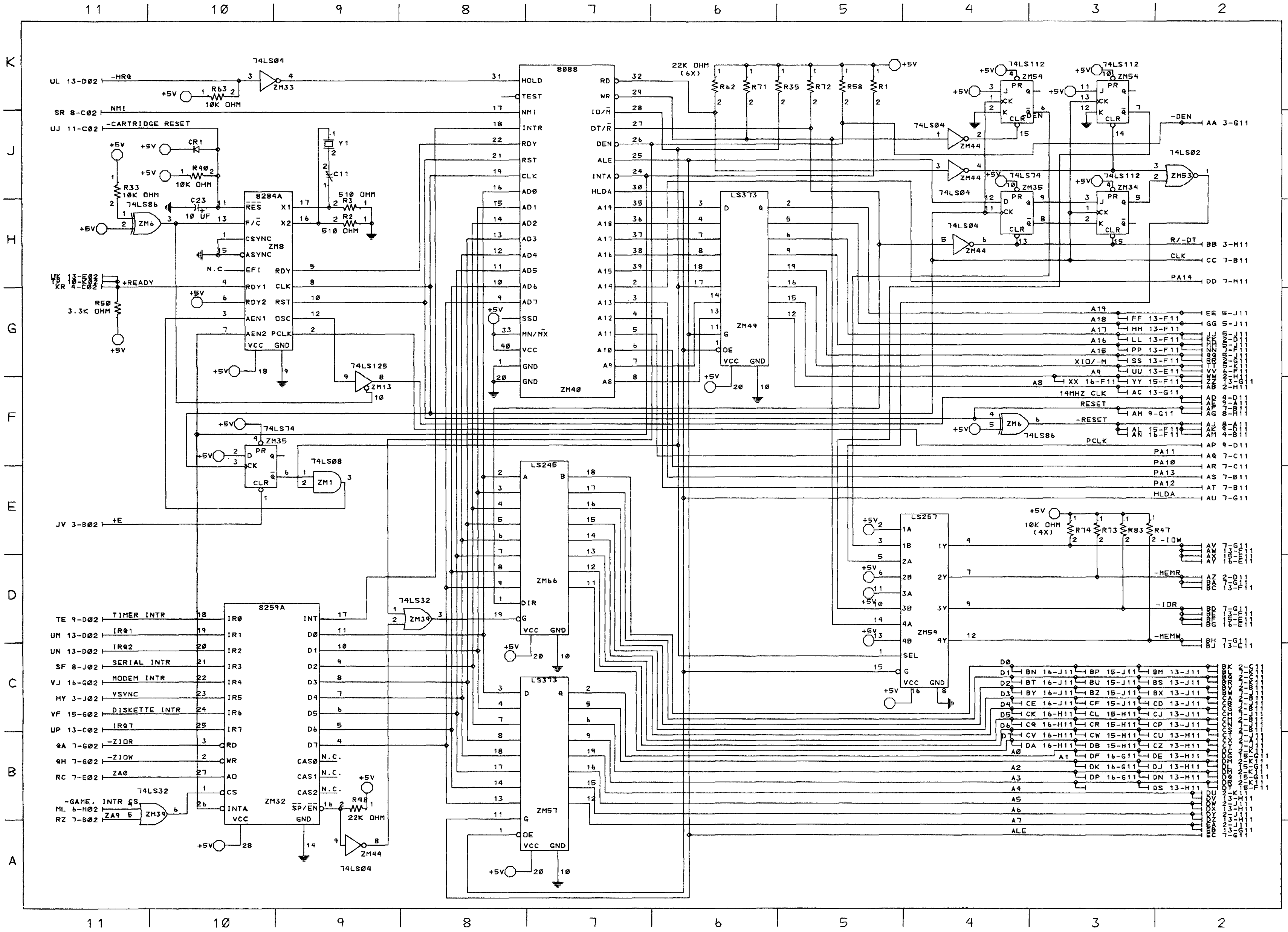
```
                                      ;-------------------------------------------;
                                      ;  EASE OF USE REVECTOR ROUTINE - CALLED THROUGH
                                      ;  INT 18H WHEN CASSETTE BASIC IS INVOKED (NO DISKETTE
                                      ;  NO CARTRIDGES)
                                      ;  KEYBOARD VECTOR IS RESET TO POINT TO "NEW_INT_9"
                                      ;  BASIC VECTOR IS SET TO POINT TO F600:0
                                      ;-------------------------------------------
FFCB                                  BAS_ENT PROC    FAR
                                              ASSUME  DS:ABSO
FFCB  2B C0                                   SUB     AX,AX
FFCD  8E D8                                   MOV     DS,AX           ;SET ADDRESSING
FFCF  C7 06 0024 R 1937 R                     MOV     WORD PTR INT_PTR+4,OFFSET NEW_INT_9
FFD5  A3 0060 R                               MOV     BASIC_PTR,AX    ; SET INT 18=F600:0
FFD8  C7 06 0062 R F600                       MOV     BASIC_PTR+2,0F600H
FFDE  CD 18                                   INT     18H             ; GO TO BASIC
FFE0                                  BAS_ENT ENDP
                                      ;------------------------------------------------------------------
                                      ;       INITIALIZE TIMER SUBROUTINE - ASSUMES BOTH THE LSB AND MSB
                                      ;       OF THE TIMER WILL BE USED.
                                      ;   CALLING PARAMETERS:
                                      ;       (AH) = TIMER #
                                      ;       (AL) = BIT PATTERN OF INITIALIZATION WORD
                                      ;       (BX) = INITIAL COUNT
                                      ;              (BH) = MSB COUNT
                                      ;              (BL) = LSB COUNT
                                      ;   ALTERS REGISTERS DX AND AL.
                                      ;------------------------------------------------------------------
FFE0                                  INIT_TIMER      PROC  NEAR
FFE0  E6 43                                   OUT     TIM_CTL,AL      ; OUTPUT INITIAL CONTROL WORD
FFE2  BA 0040                                 MOV     DX,TIMER        ; BASE PORT ADDR FOR TIMERS
FFE5  02 D4                                   ADD     DL,AH           ; ADD IN THE TIMER #
FFE7  8A C3                                   MOV     AL,BL           ; LOAD LSB
FFE9  EE                                      OUT     DX,AL
FFEA  52                                      PUSH    DX              ; PAUSE
FFEB  5A                                      POP     DX
FFEC  8A C7                                   MOV     AL,BH           ; LOAD MSB
FFEE  EE                                      OUT     DX,AL
FFEF  C3                                      RET
FFF0                                  INIT_TIMER      ENDP

                                      ;--------------------------------
                                      ;       POWER ON RESET VECTOR   :
                                      ;--------------------------------
FFF0                                          ORG     0FFF0H
                                      ;
                                      ;----- POWER ON RESET
                                      ;
FFF0  EA                                      DB      0EAH                    ; JUMP FAR
FFF1  0043 R                                  DW      OFFSET RESET
FFF3  F000                                    DW      0F000H

FFF5  30 36 2F 30 31 2F                       DB      '06/01/83'              ; RELEASE MARKER
      38 33

FFFD  FF                                      DB      0FFH                    ; FILLER

FFFE  FD                                      DB      0FDH                    ; SYSTEM IDENTIFIER

                                      ;       DB      0FFH                    ; CHECKSUM
FFFF                                  CODE    ENDS
                                              END
```

Notes:

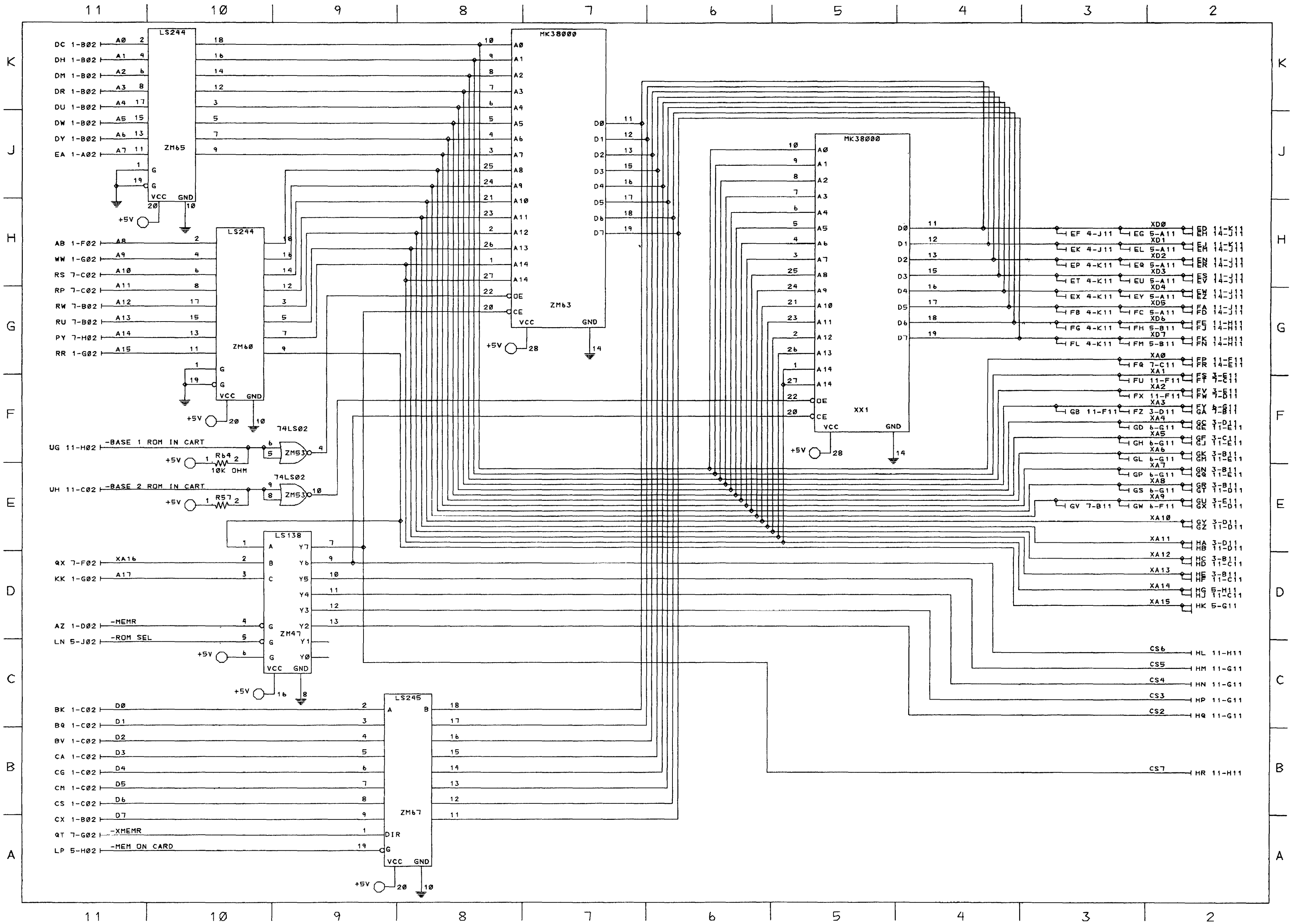
Appendix B. LOGIC DIAGRAMS

Contents

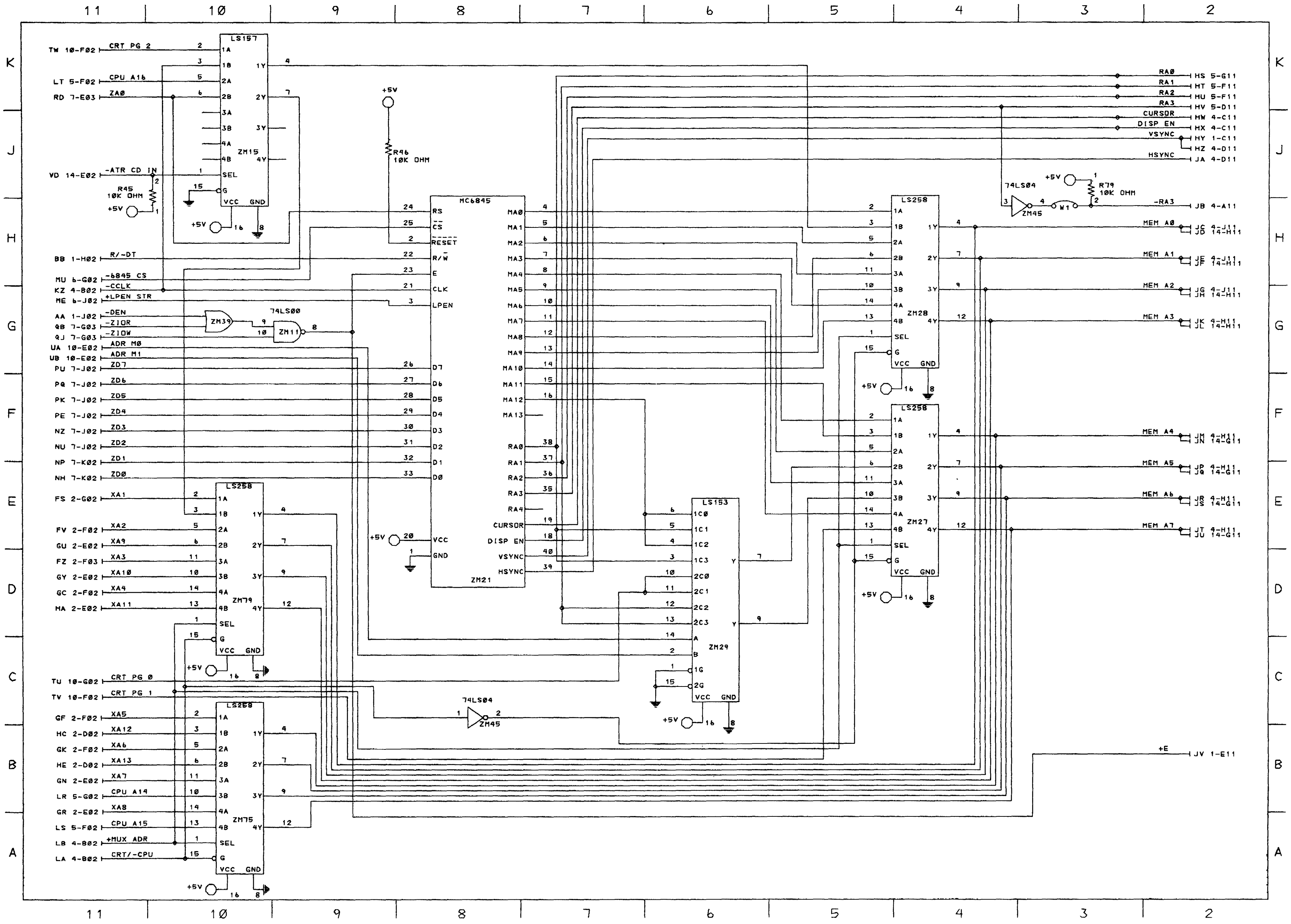
Notes:



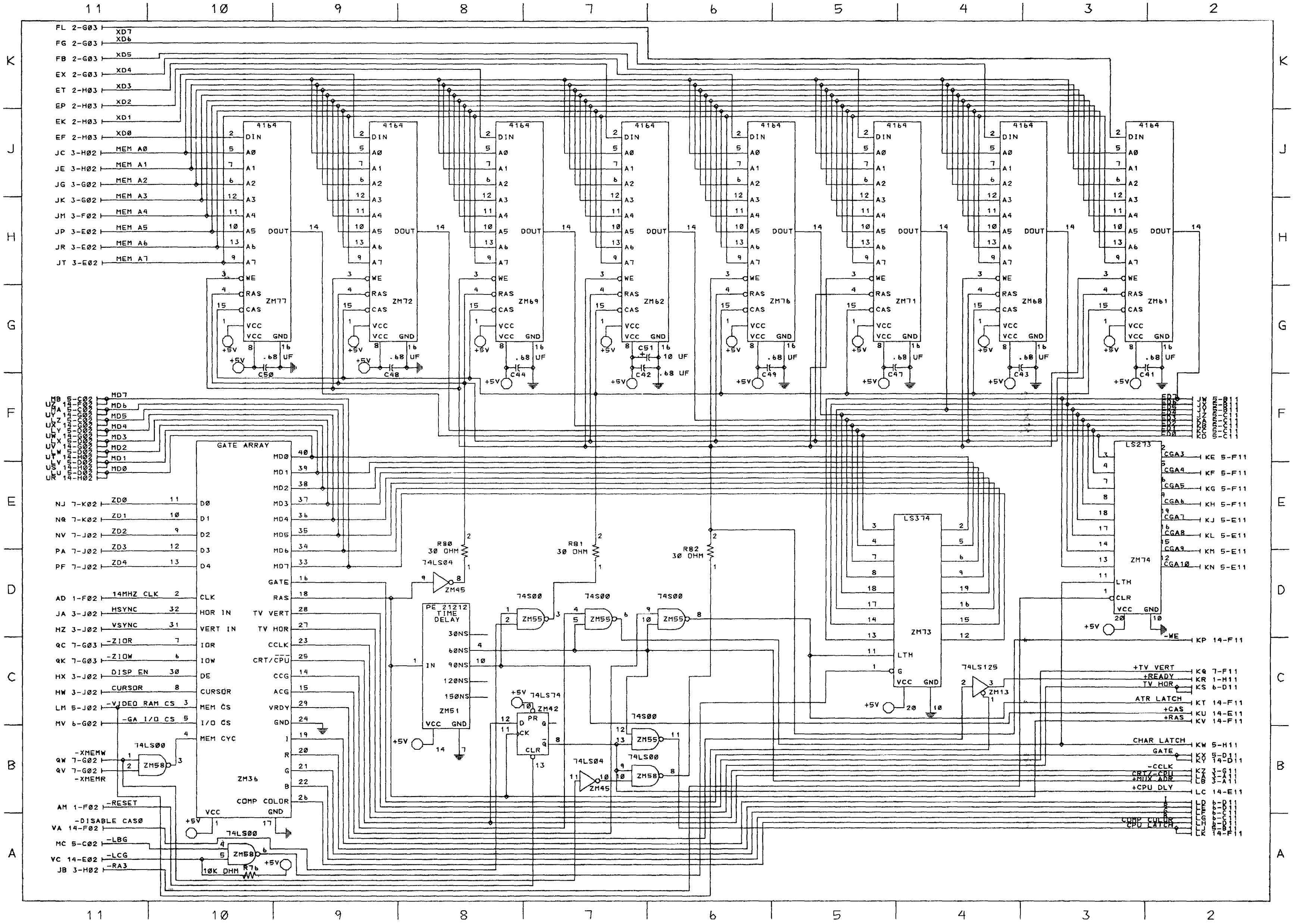
System Board (Sheet 1 of 17)



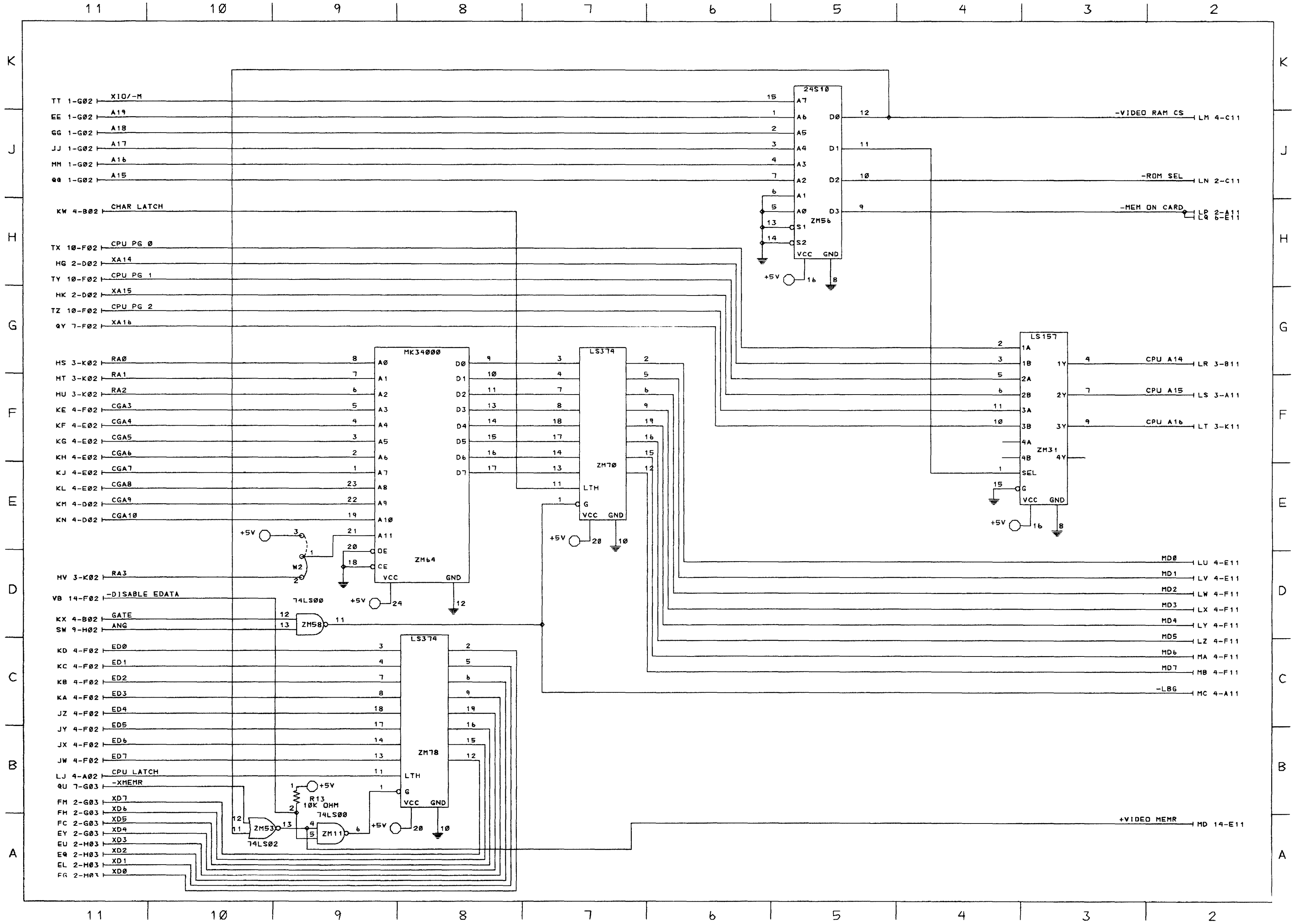
System Board (Sheet 2 of 17)



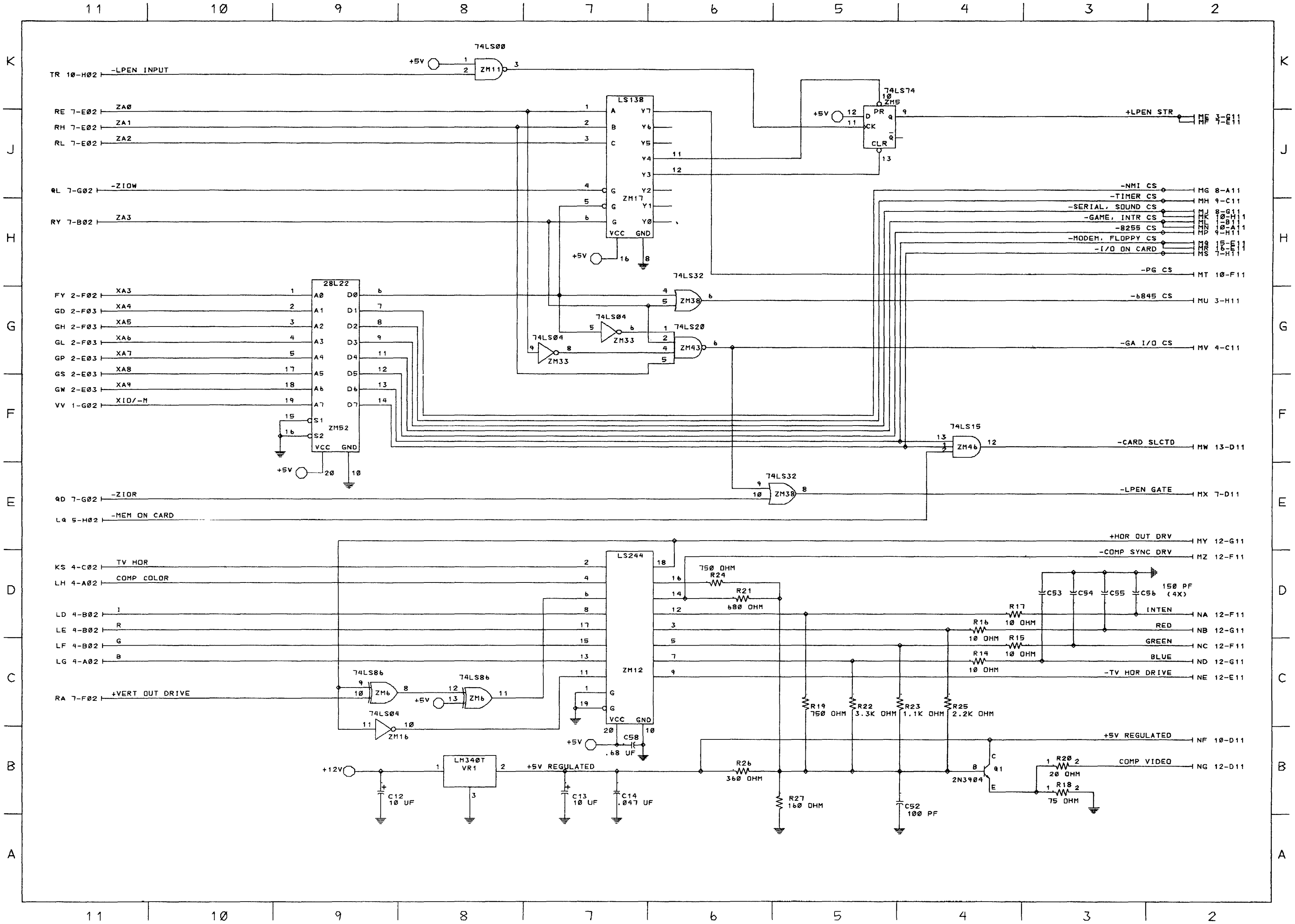
System Board (Sheet 3 of 17)



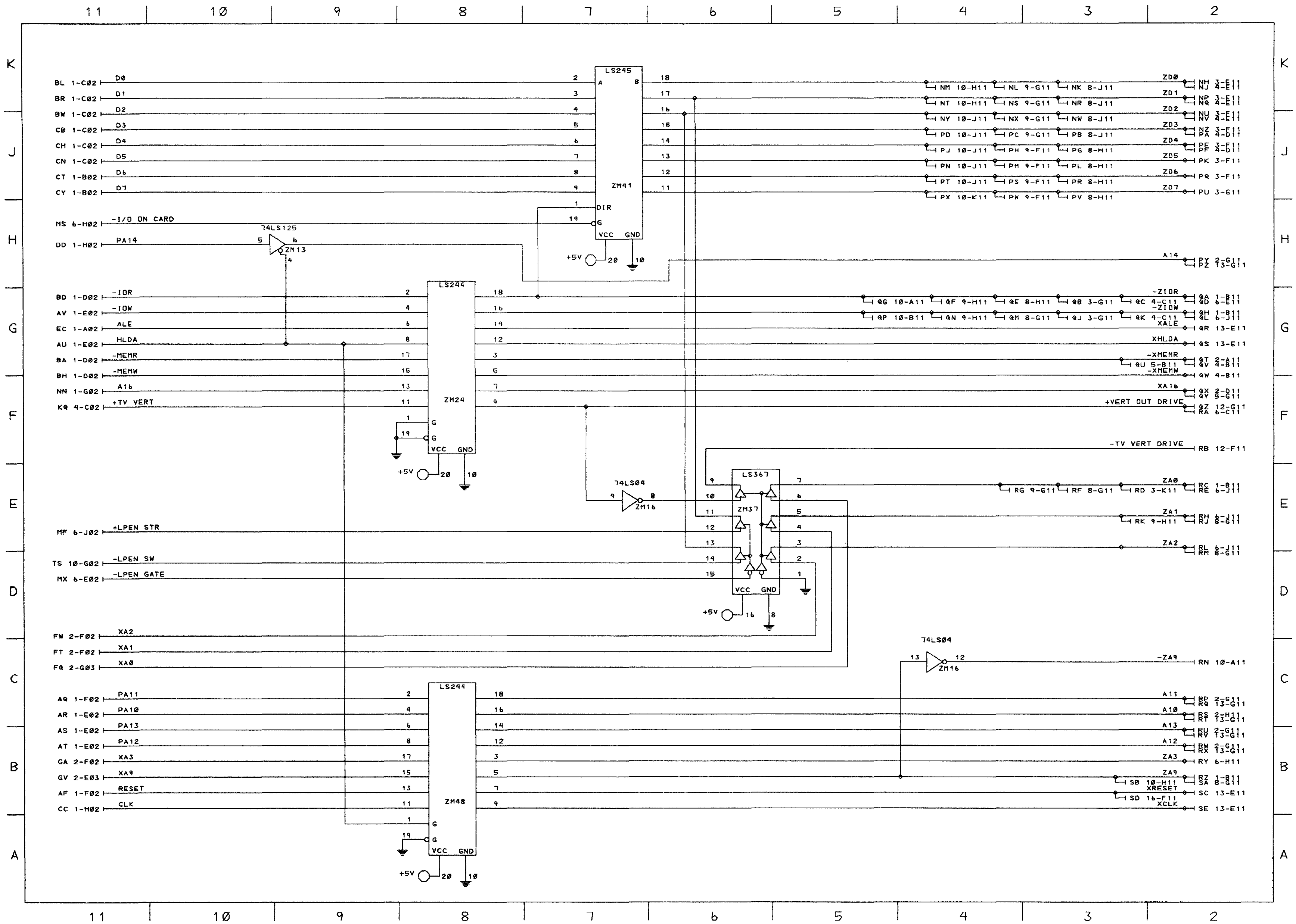
System Board (Sheet 4 of 17)



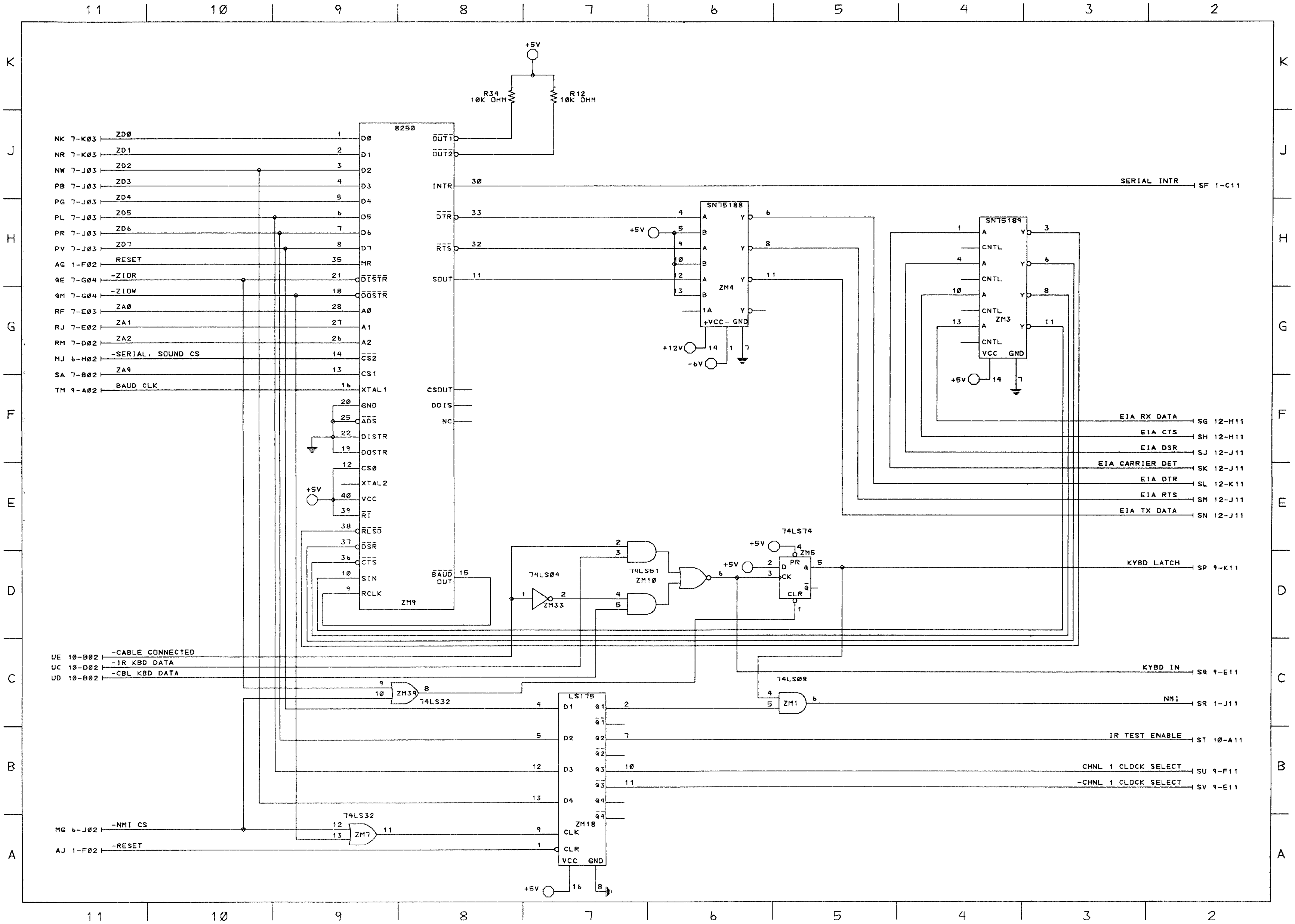
System Board (Sheet 5 of 17)



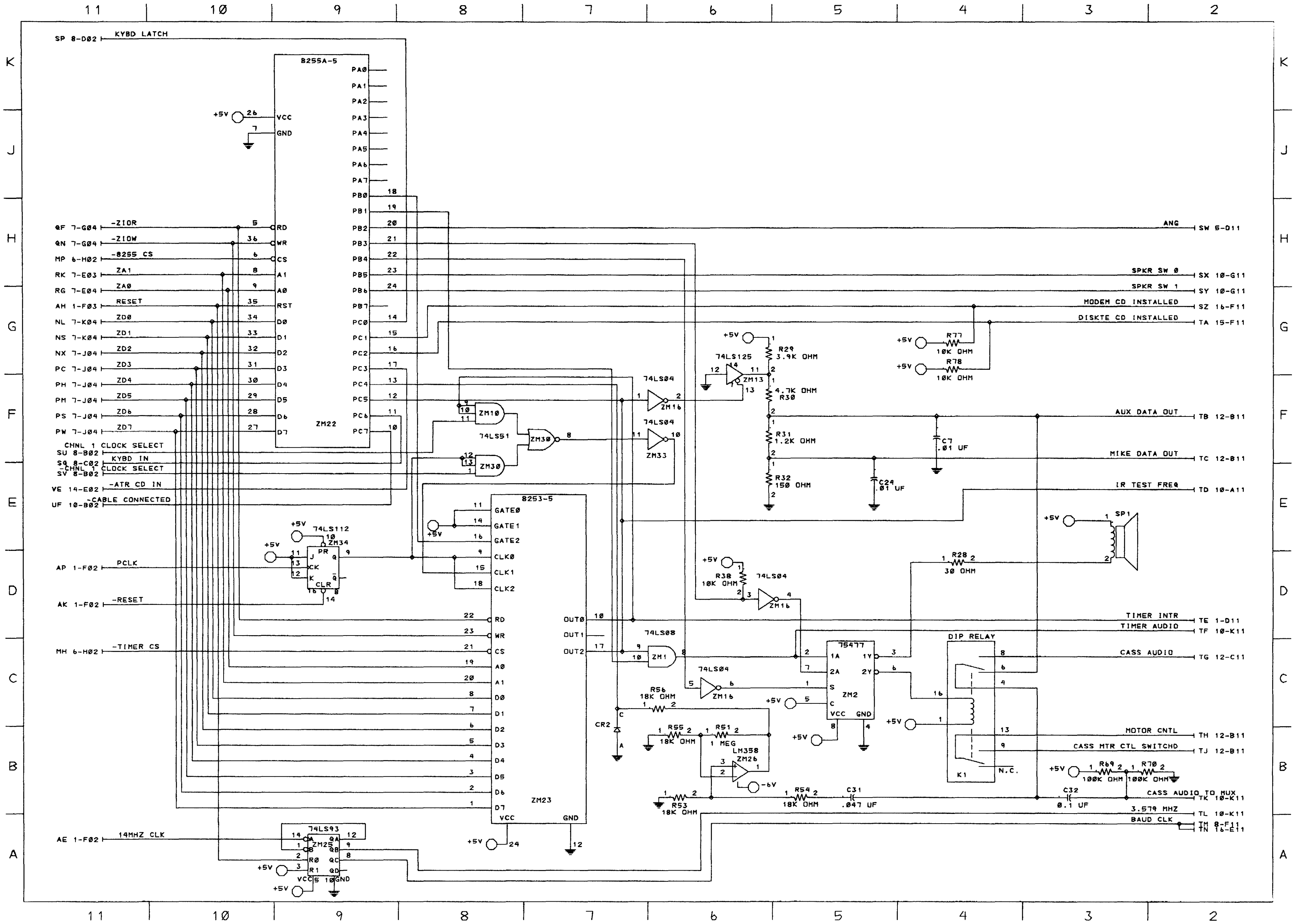
System Board (Sheet 6 of 17)



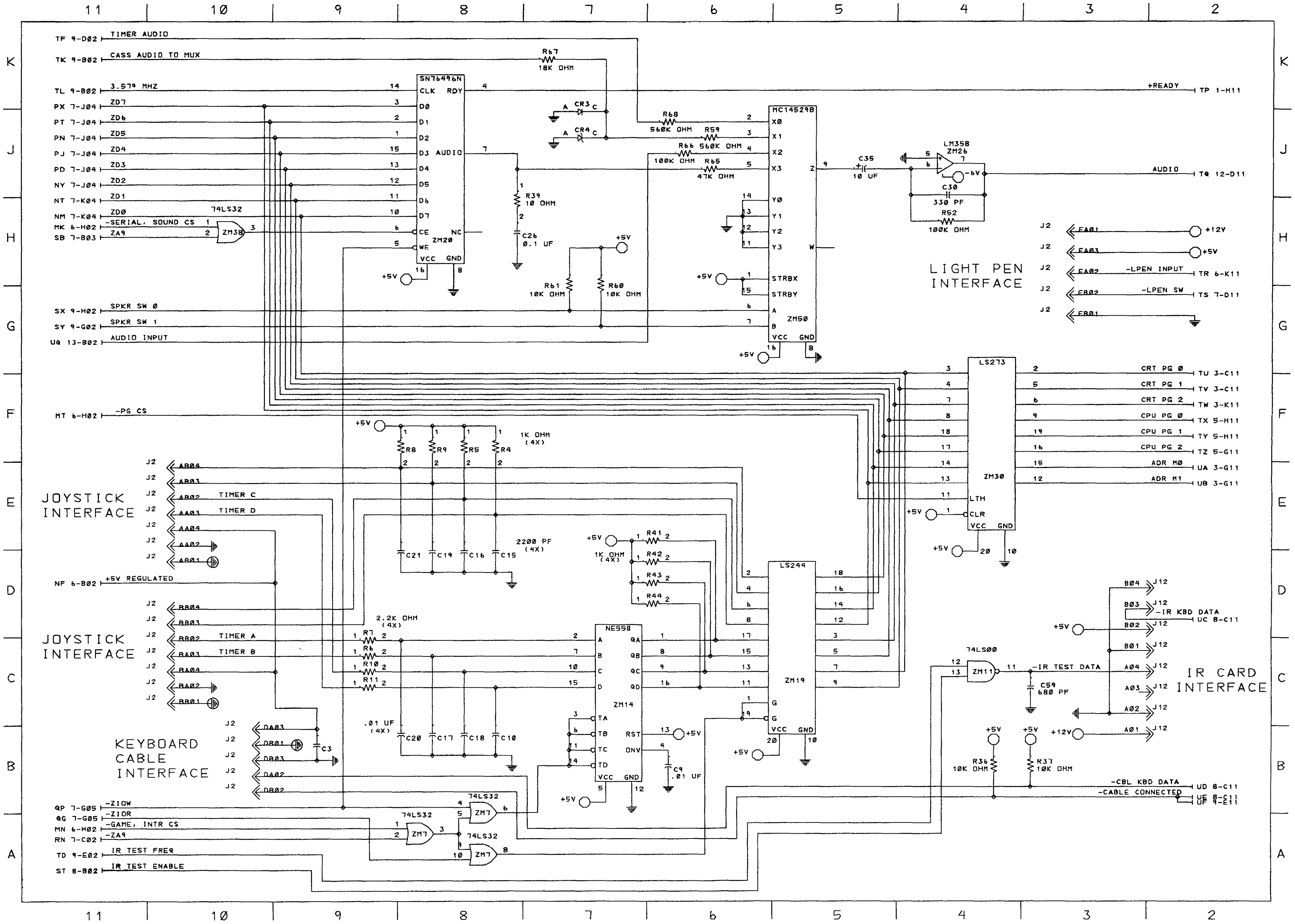
System Board (Sheet 7 of 17)



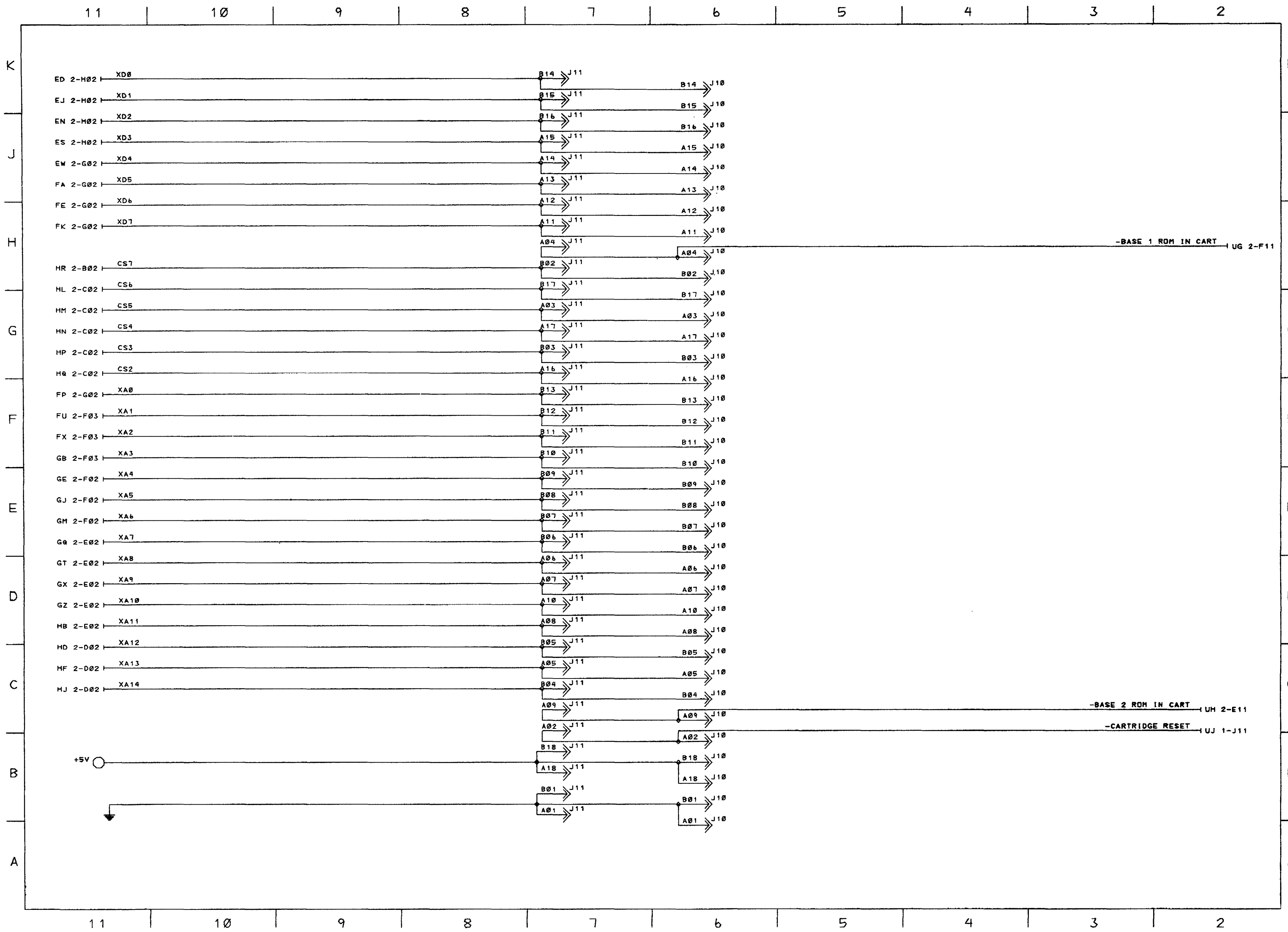
System Board (Sheet 8 of 17)



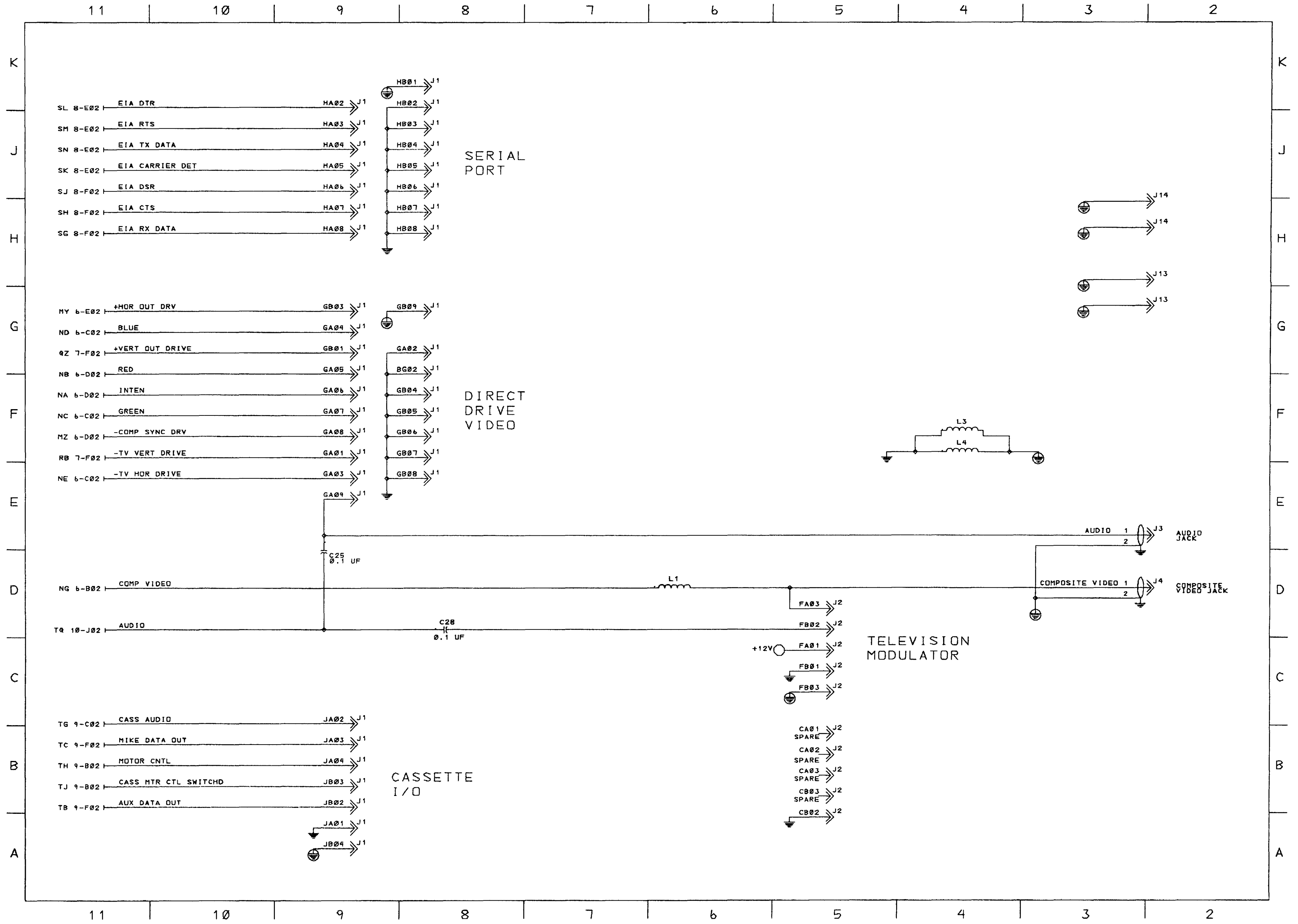
System Board (Sheet 9 of 17)



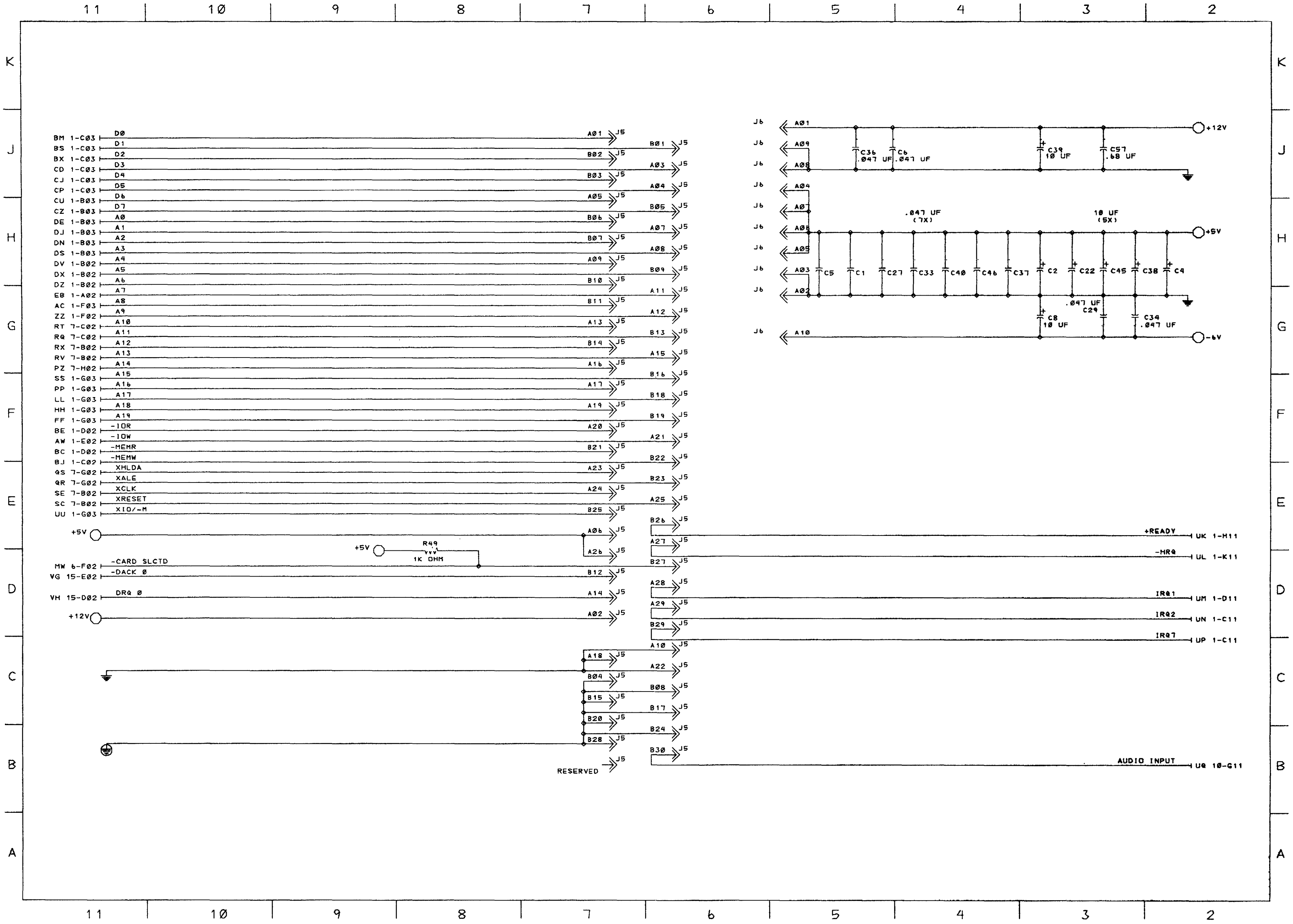
System Board (Sheet 10 of 17)


11 10 9 8 7 6 5 4 3 2
K J H G F E D C B A
ED 2-HØ2 XDØ B14 J11 B14 J10
EJ 2-HØ2 XD1 B15 J11 B15 J10
EN 2-HØ2 XD2 B16 J11 B16 J10
ES 2-HØ2 XD3 A15 J11 A15 J10
EW 2-GØ2 XD4 A14 J11 A14 J10
FA 2-GØ2 XD5 A13 J11 A13 J10
FE 2-GØ2 XD6 A12 J11 A12 J10
FK 2-GØ2 XD7 A11 J11 A11 J10
AØ4 J11 AØ4 J10
-BASE 1 ROM IN CART UG 2-F11
HR 2-BØ2 CS7 BØ2 J11 BØ2 J10
HL 2-CØ2 CS6 B17 J11 B17 J10
HM 2-CØ2 CS5 AØ3 J11 AØ3 J10
HN 2-CØ2 CS4 A17 J11 A17 J10
HP 2-CØ2 CS3 BØ3 J11 BØ3 J10
HQ 2-CØ2 CS2 A16 J11 A16 J10
FP 2-GØ2 XAØ B13 J11 B13 J10
FU 2-FØ3 XA1 B12 J11 B12 J10
FX 2-FØ3 XA2 B11 J11 B11 J10
GB 2-FØ3 XA3 B1Ø J11 B1Ø J10
GE 2-FØ2 XA4 BØ9 J11 BØ9 J10
GJ 2-FØ2 XA5 BØ8 J11 BØ8 J10
GM 2-FØ2 XA6 BØ7 J11 BØ7 J10
GQ 2-EØ2 XA7 BØ6 J11 BØ6 J10
GT 2-EØ2 XA8 AØ6 J11 AØ6 J10
GX 2-EØ2 XA9 AØ7 J11 AØ7 J10
GZ 2-EØ2 XA1Ø A1Ø J11 A1Ø J10
HB 2-EØ2 XA11 AØ8 J11 AØ8 J10
HD 2-DØ2 XA12 BØ5 J11 BØ5 J10
HF 2-DØ2 XA13 AØ5 J11 AØ5 J10
HJ 2-DØ2 XA14 BØ4 J11 BØ4 J10
AØ9 J11 AØ9 J10
-BASE 2 ROM IN CART UH 2-E11
AØ2 J11 AØ2 J10
-CARTRIDGE RESET UJ 1-J11
+5V
B18 J11 B18 J10
A18 J11 A18 J10
BØ1 J11 BØ1 J10
AØ1 J11 AØ1 J10

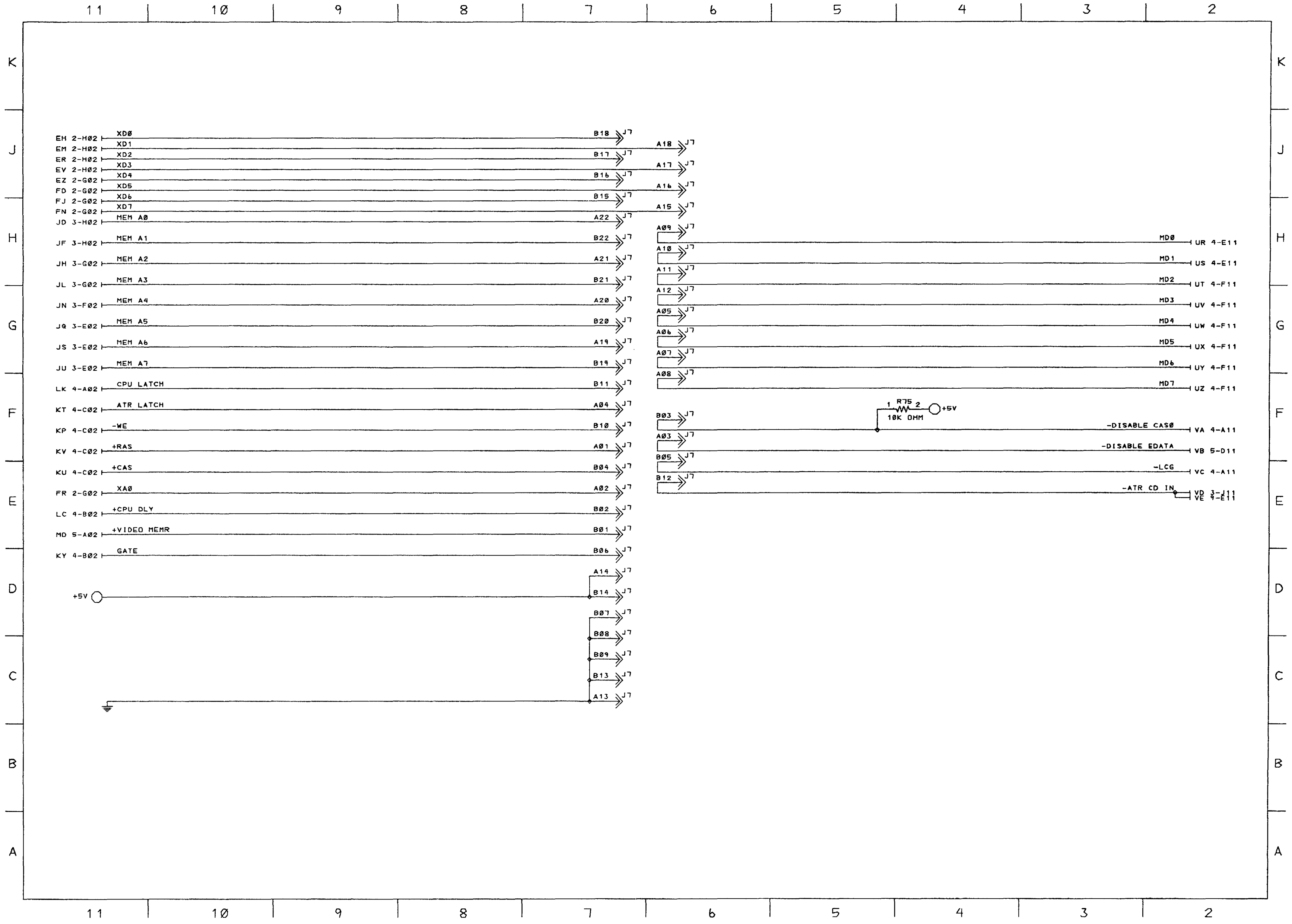
System Board (Sheet 11 of 17)



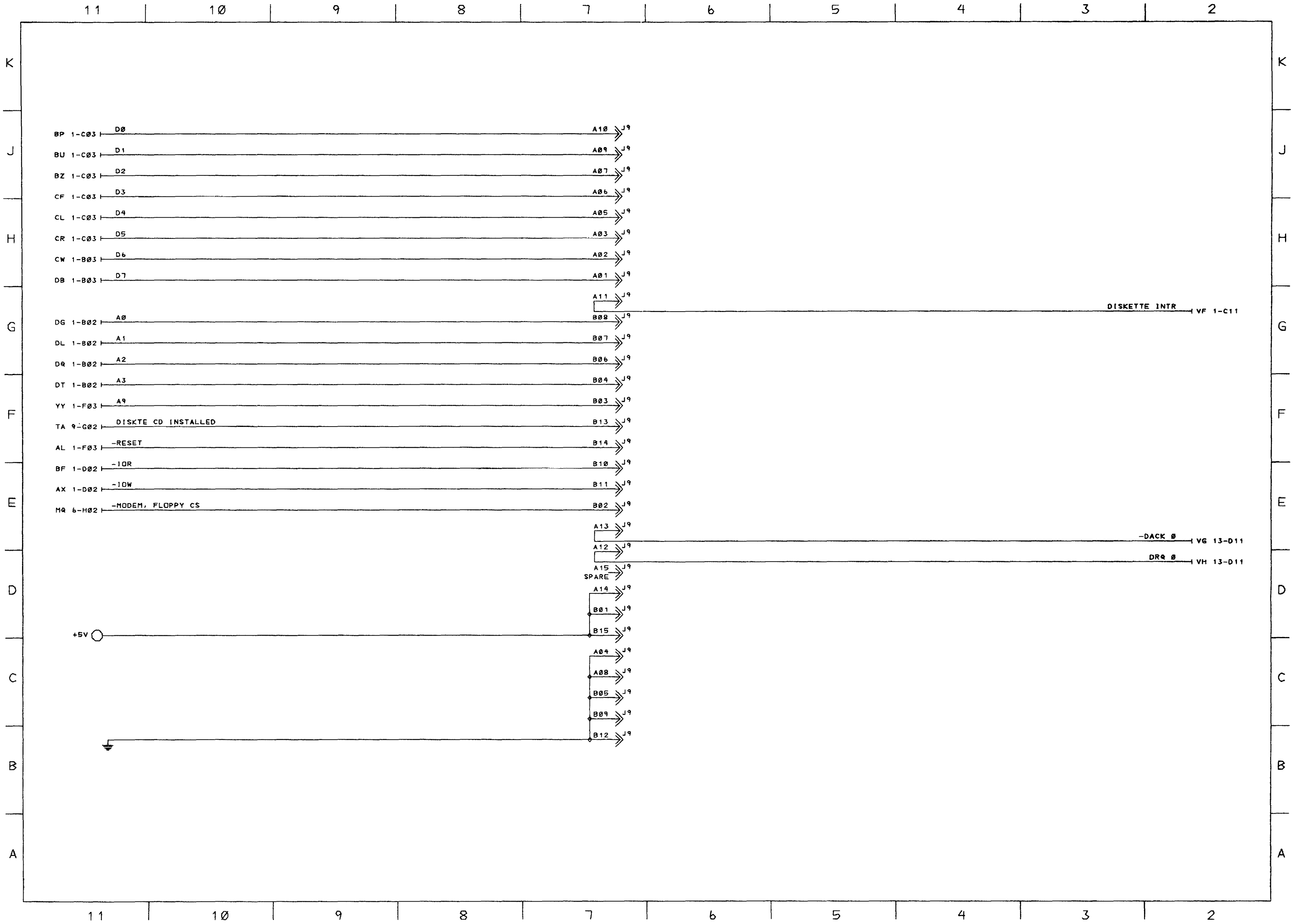
System Board (Sheet 12 of 17)



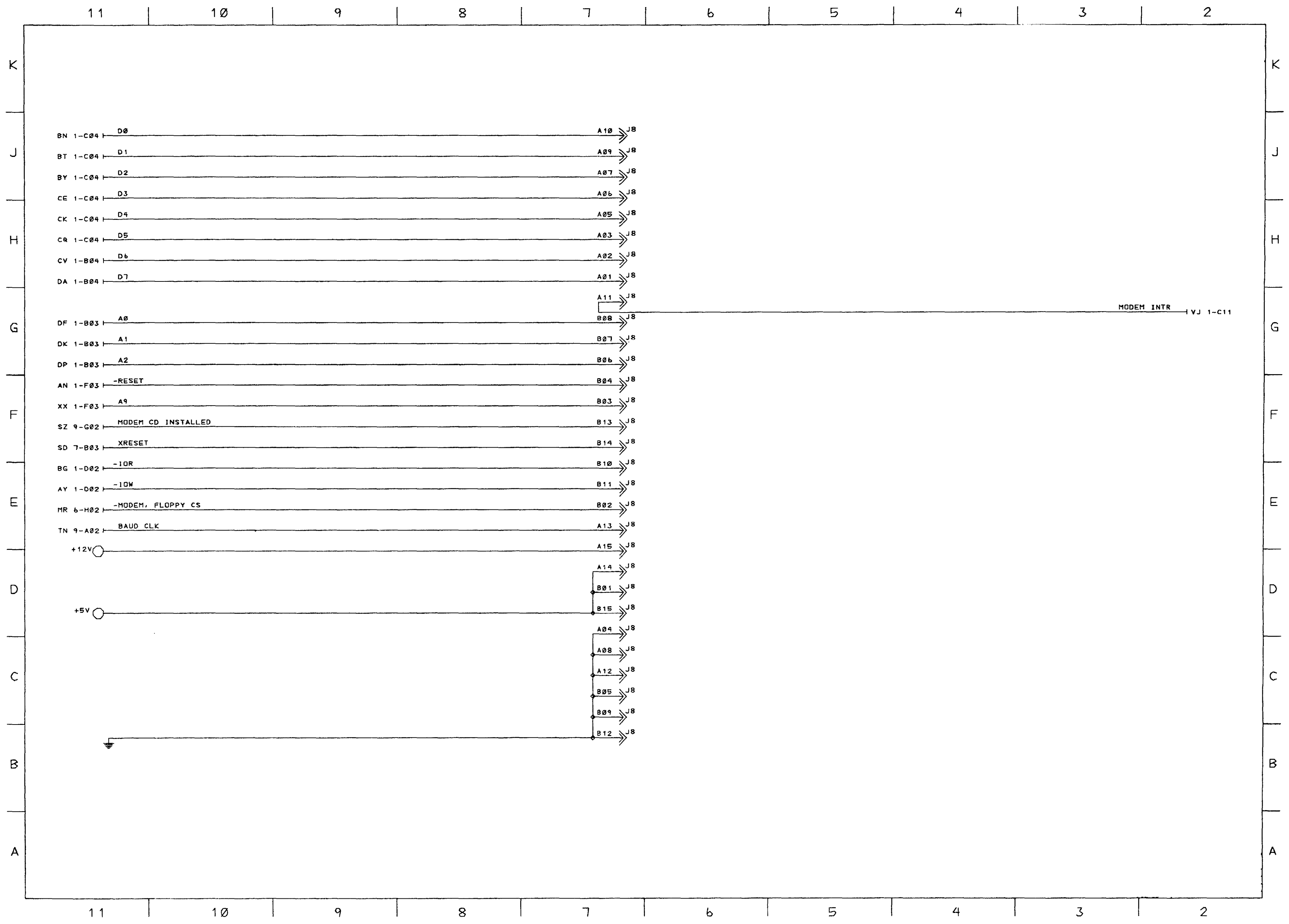
System Board (Sheet 13 of 17)



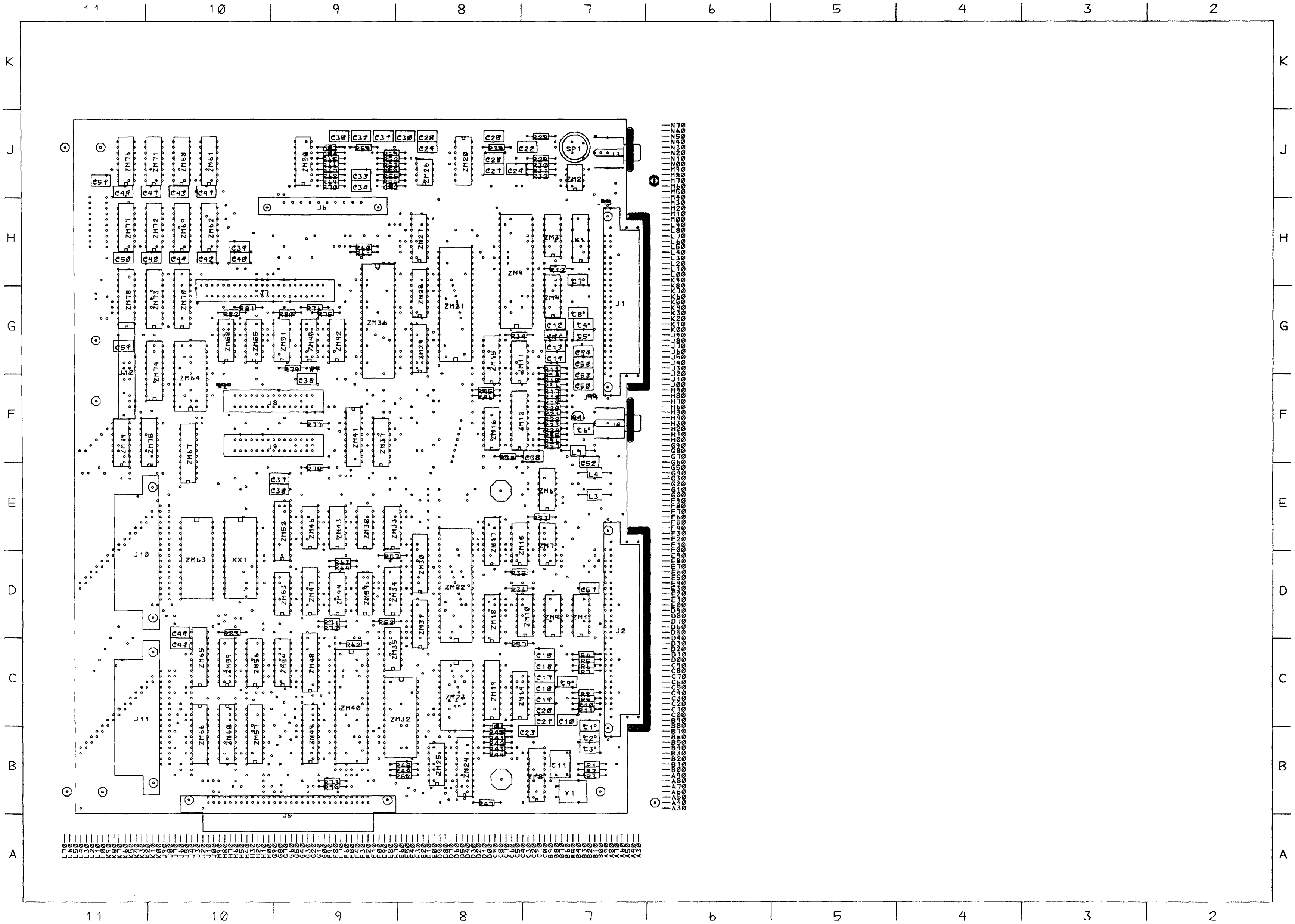
System Board (Sheet 14 of 17)



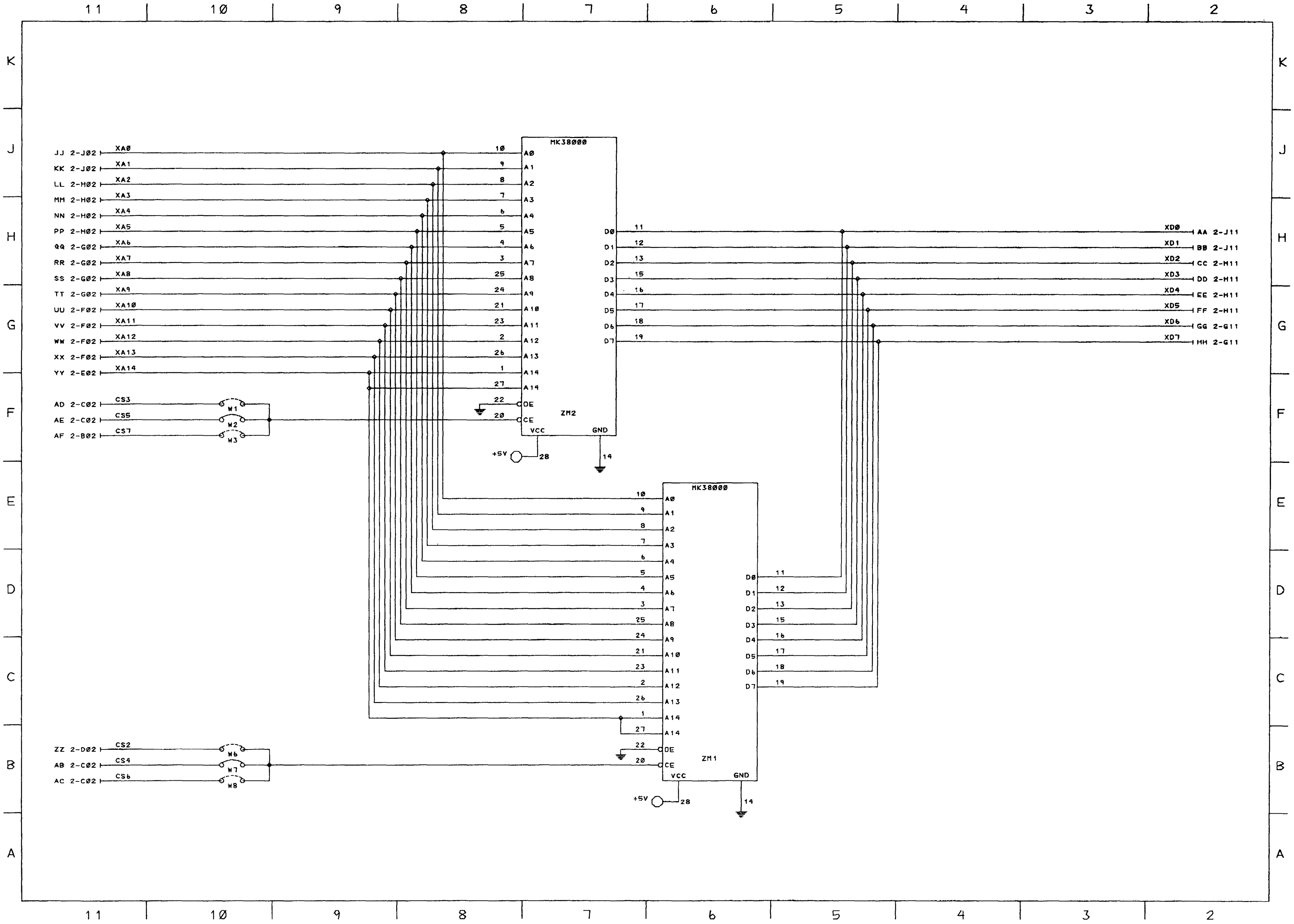
System Board (Sheet 15 of 17)



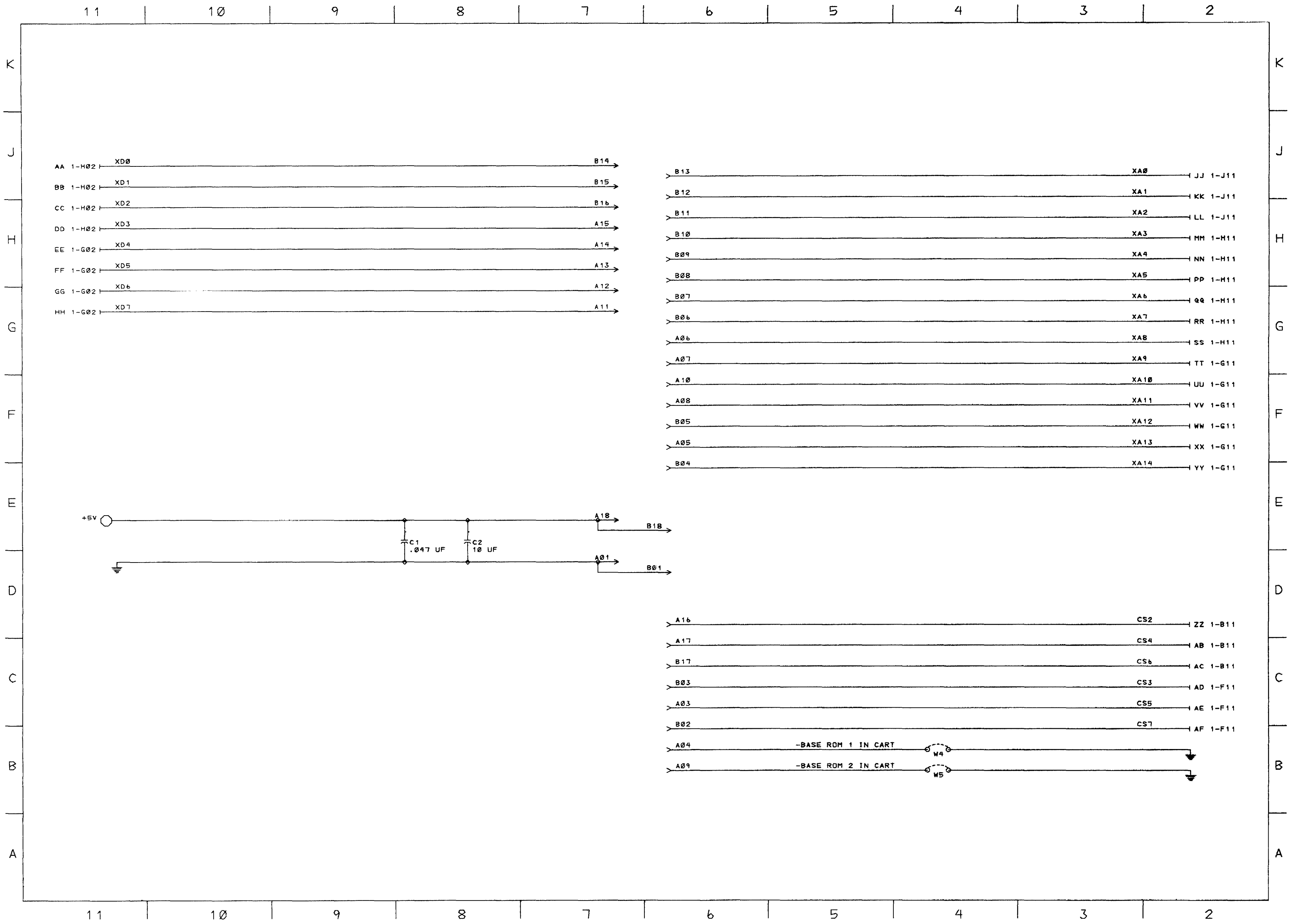
System Board (Sheet 16 of 17)



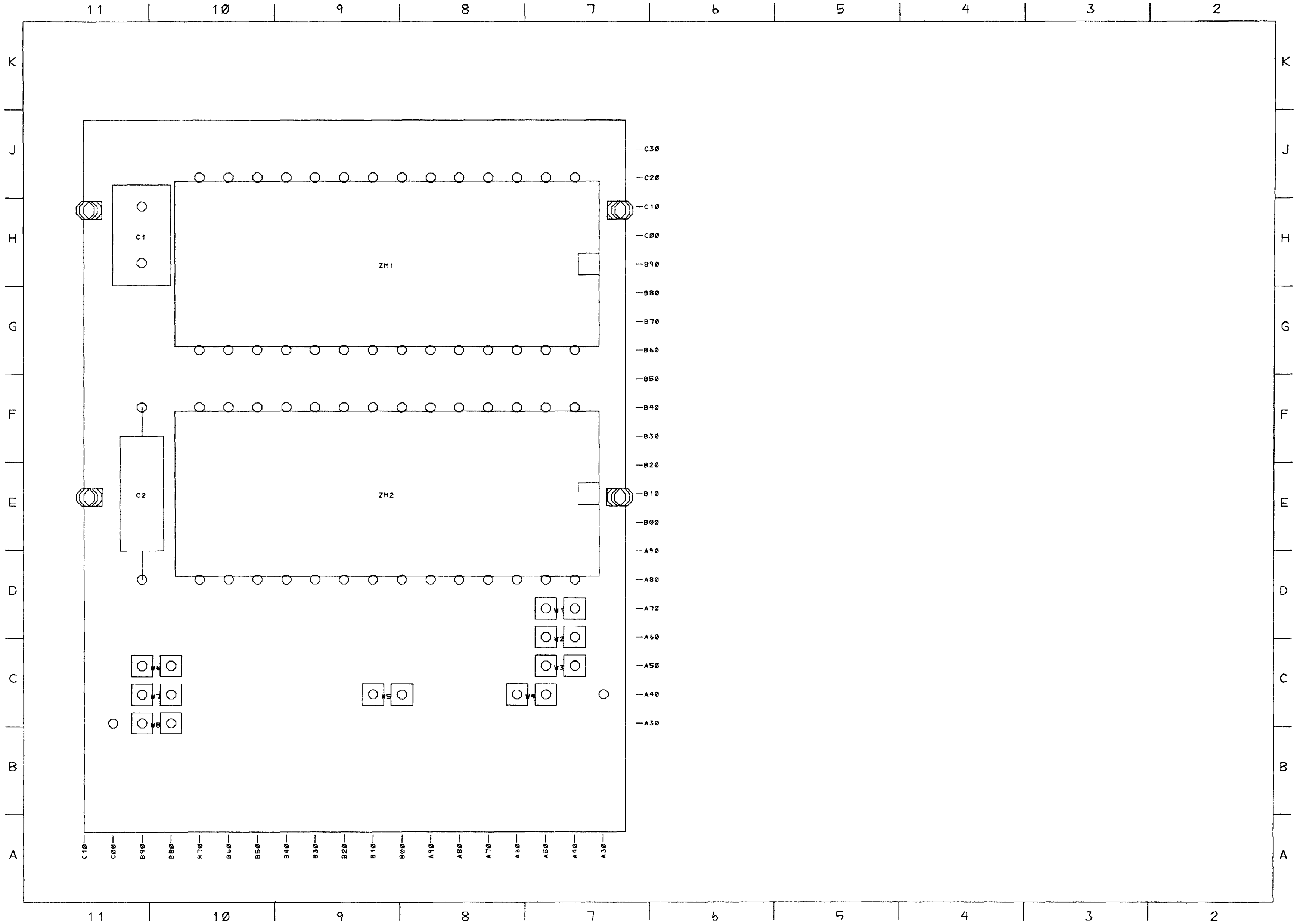
System Board (Sheet 17 of 17)



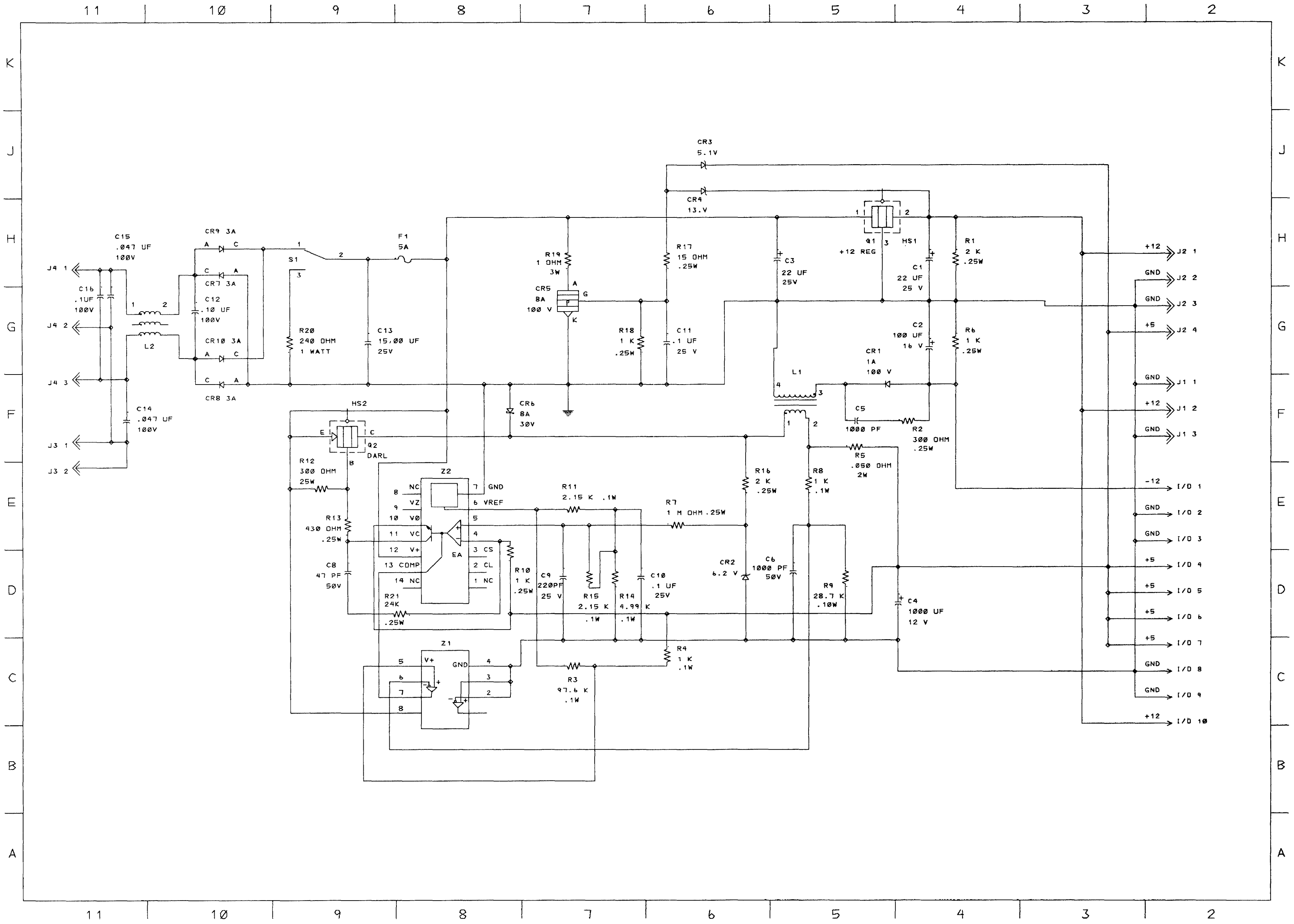
Program Cartridge (Sheet 1 of 3)



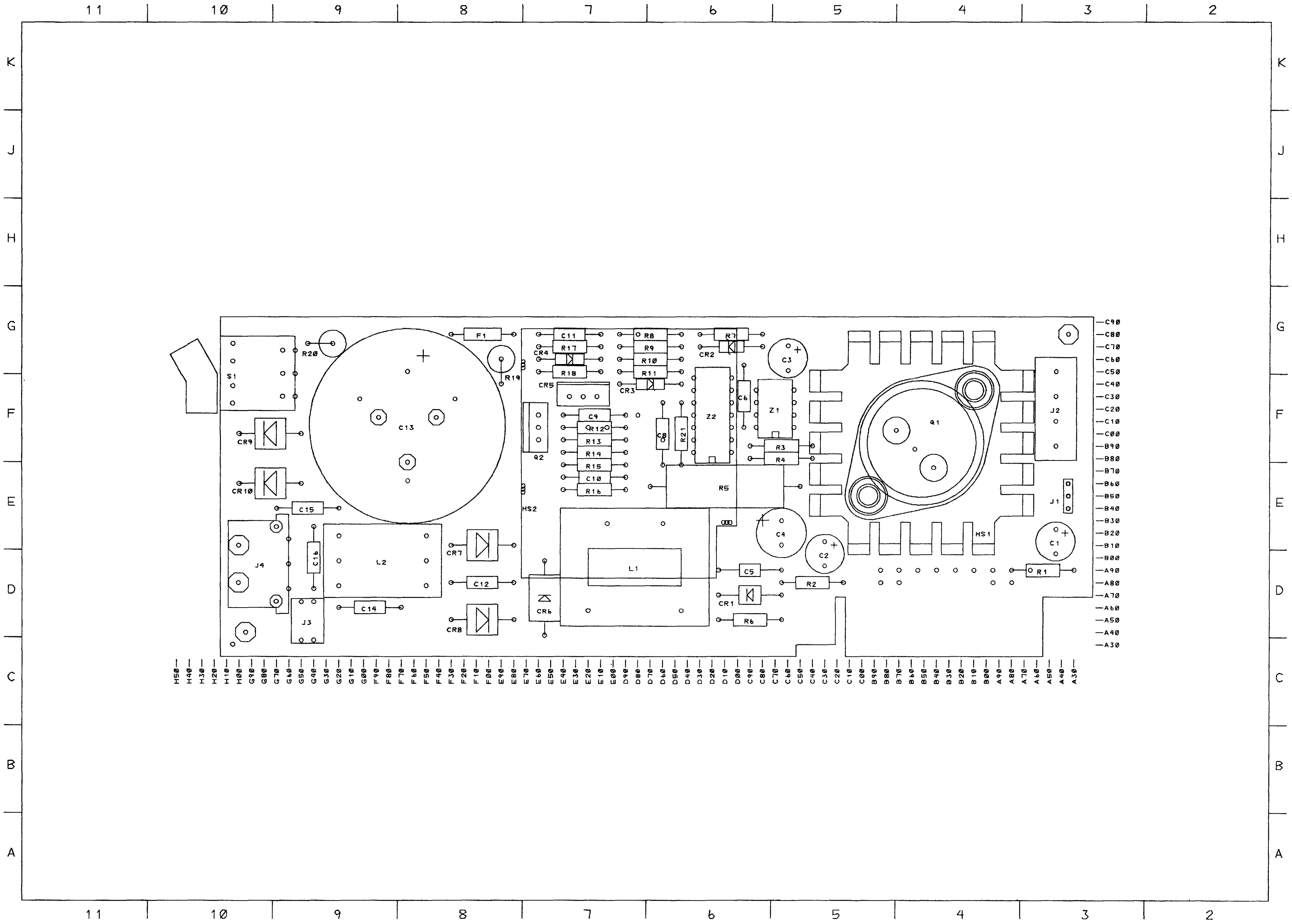
Program Cartridge (Sheet 2 of 3)



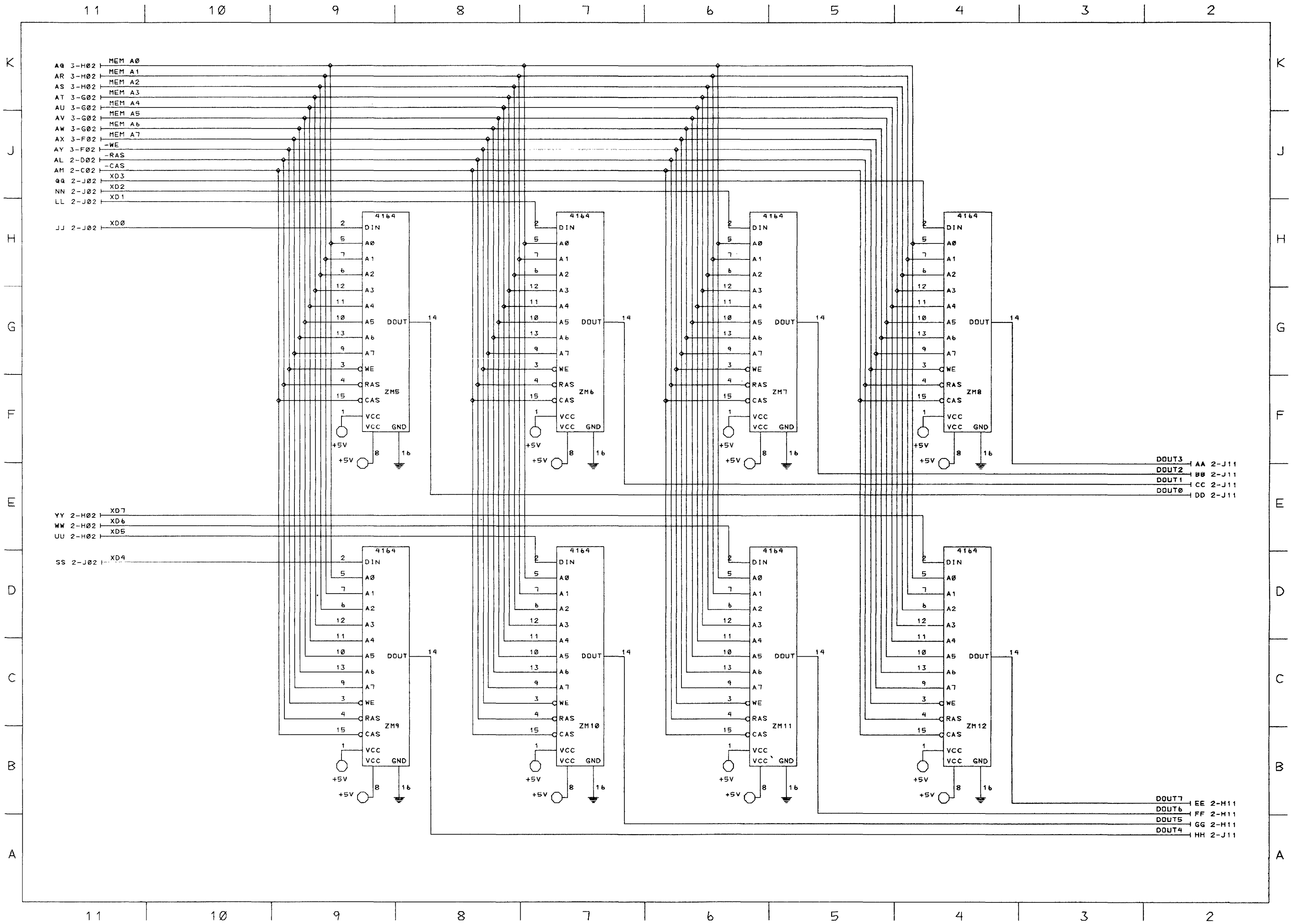
Program Cartridge (Sheet 3 of 3)



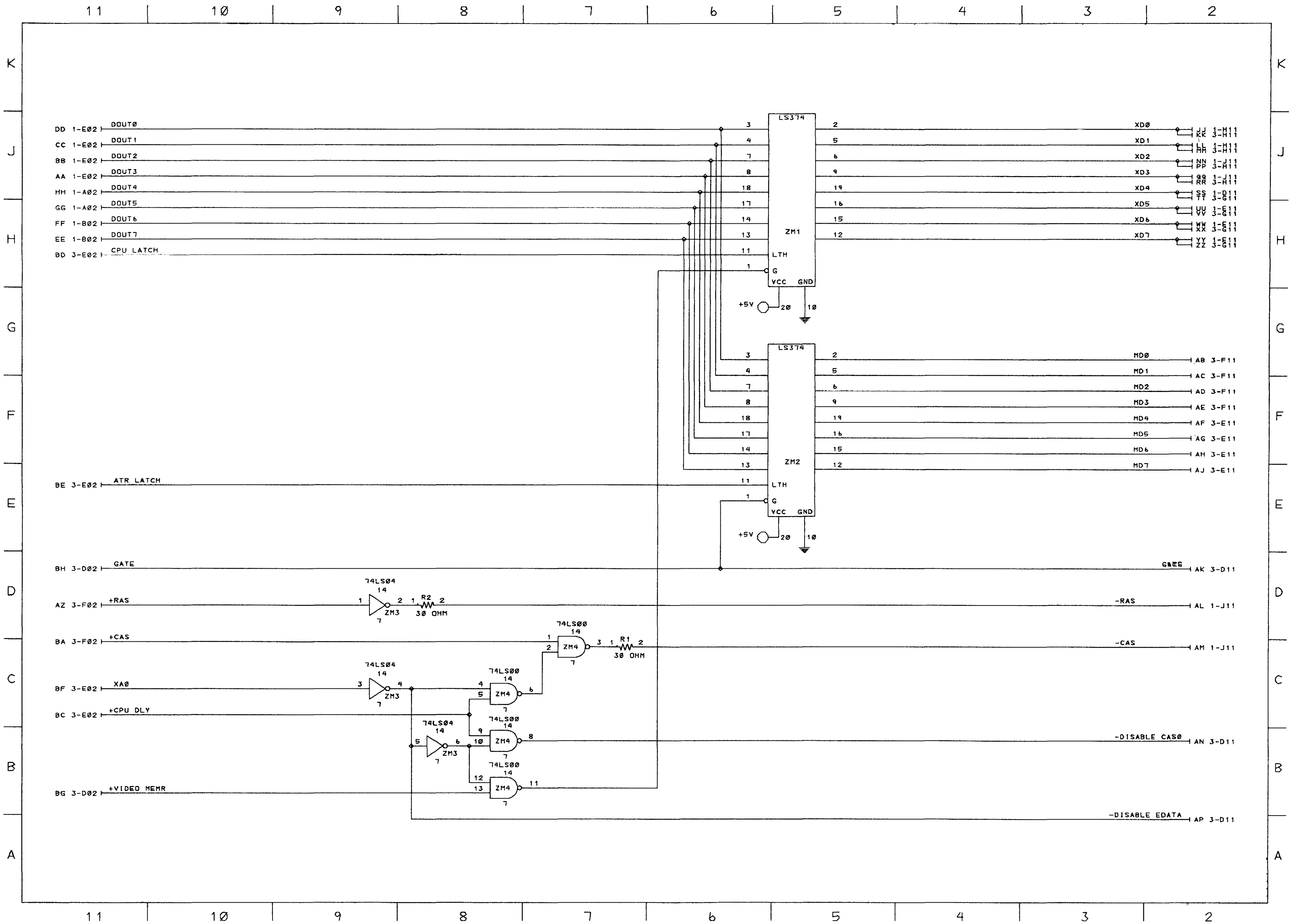
Power Supply Board (Sheet 1 of 2)



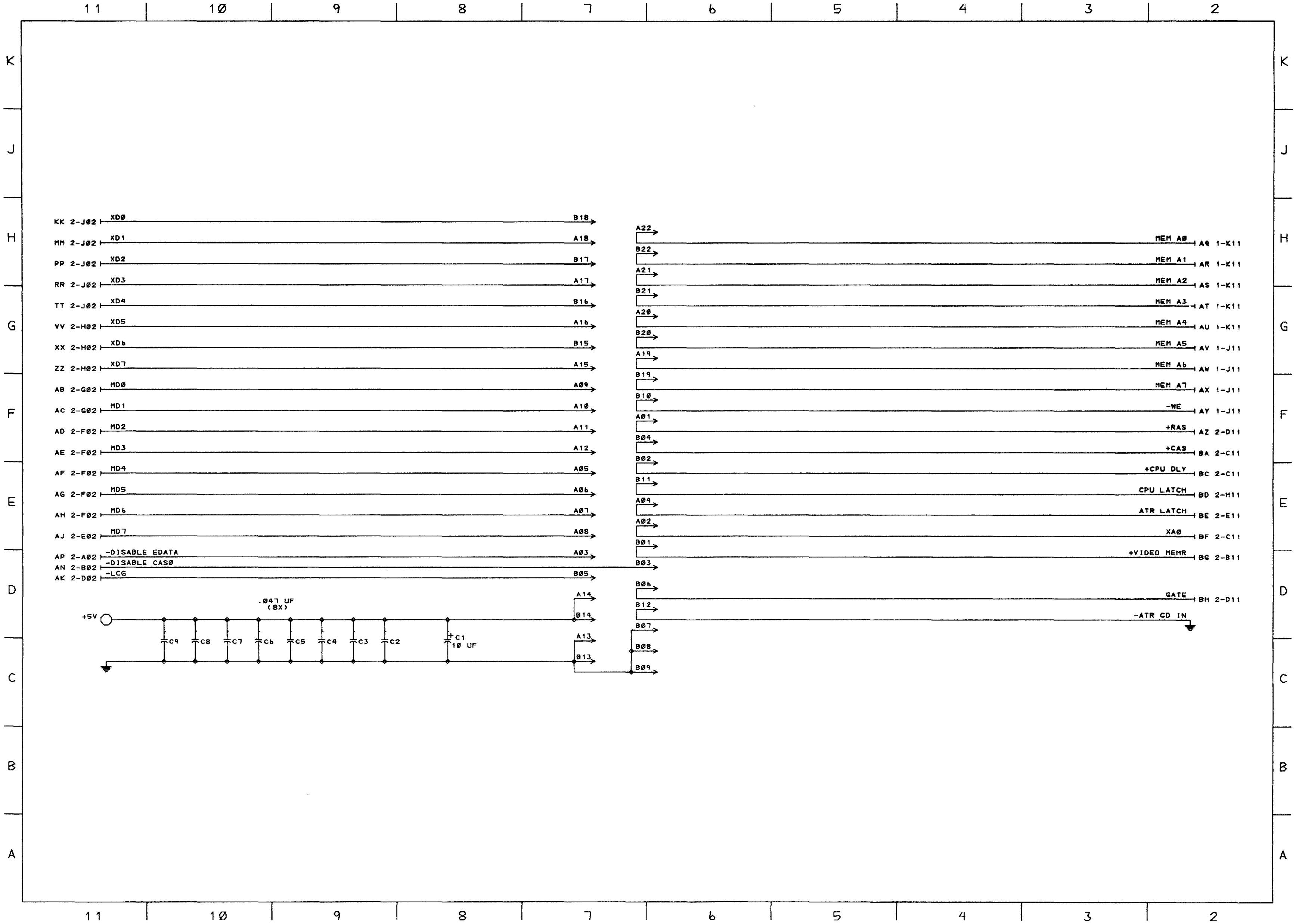
Power Supply Board (Sheet 2 of 2)



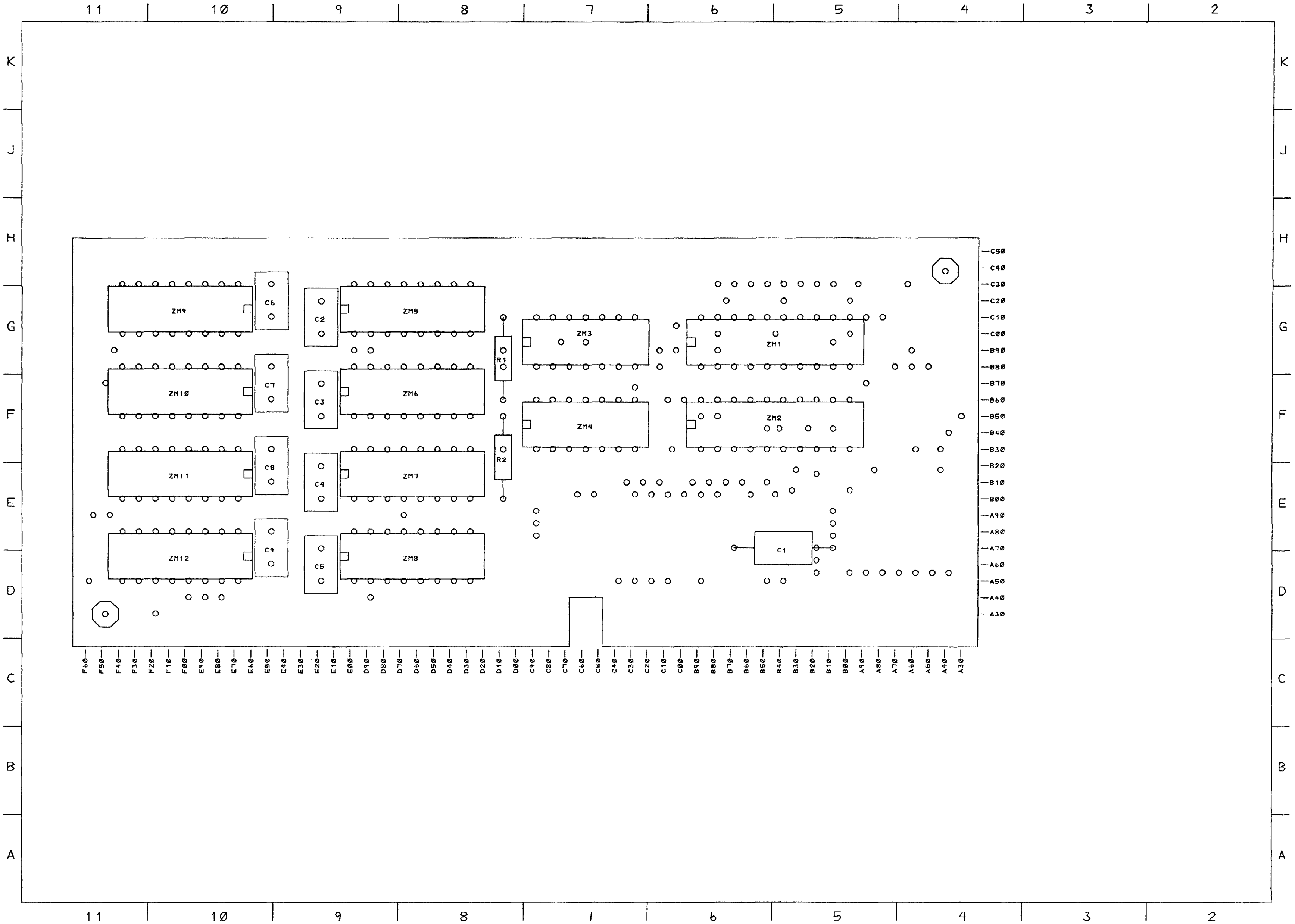
64KB Memory and Display Expansion (Sheet 1 of 4)



64KB Memory and Display Expansion (Sheet 2 of 4)



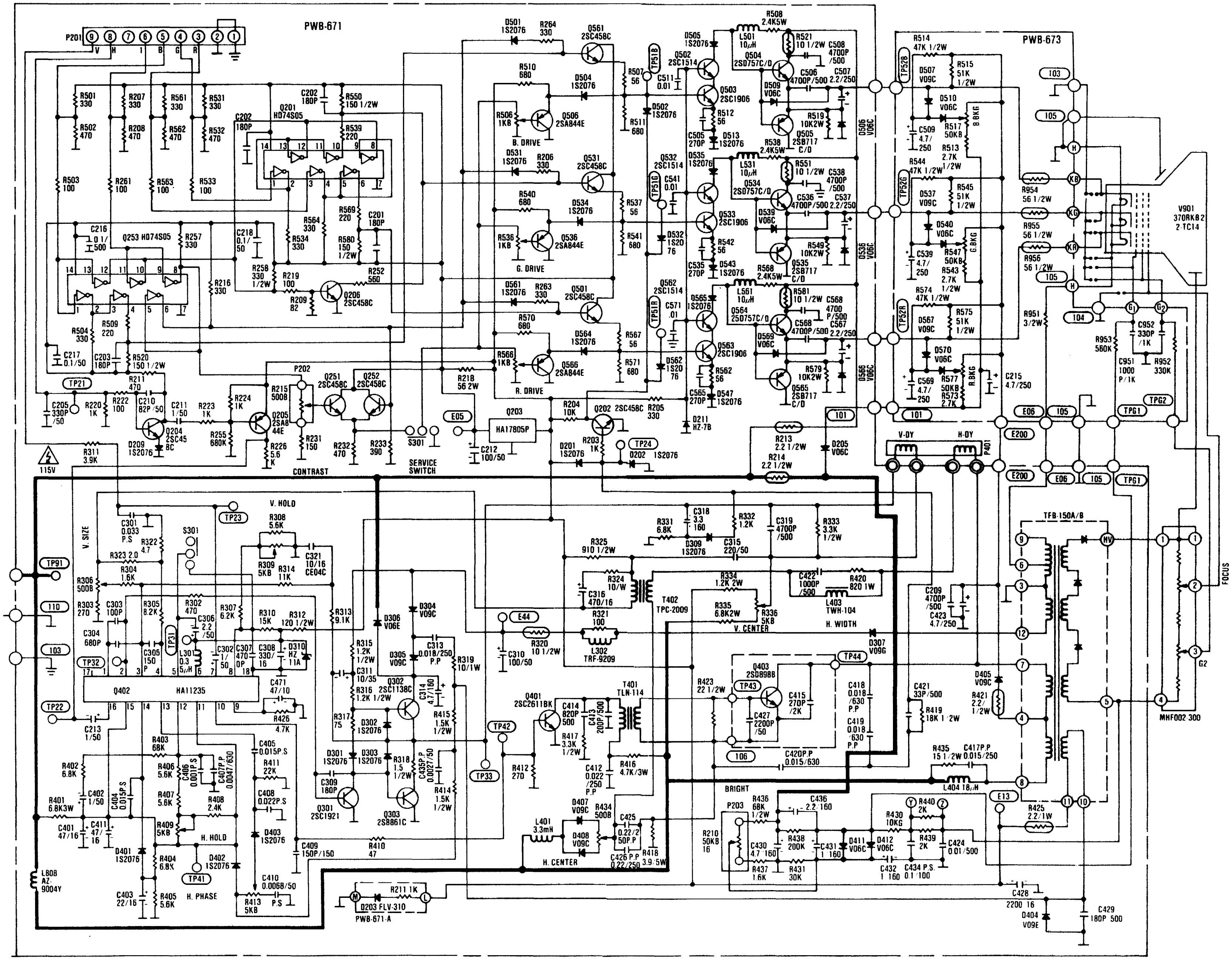
64KB Memory and Display Expansion (Sheet 3 of 4)



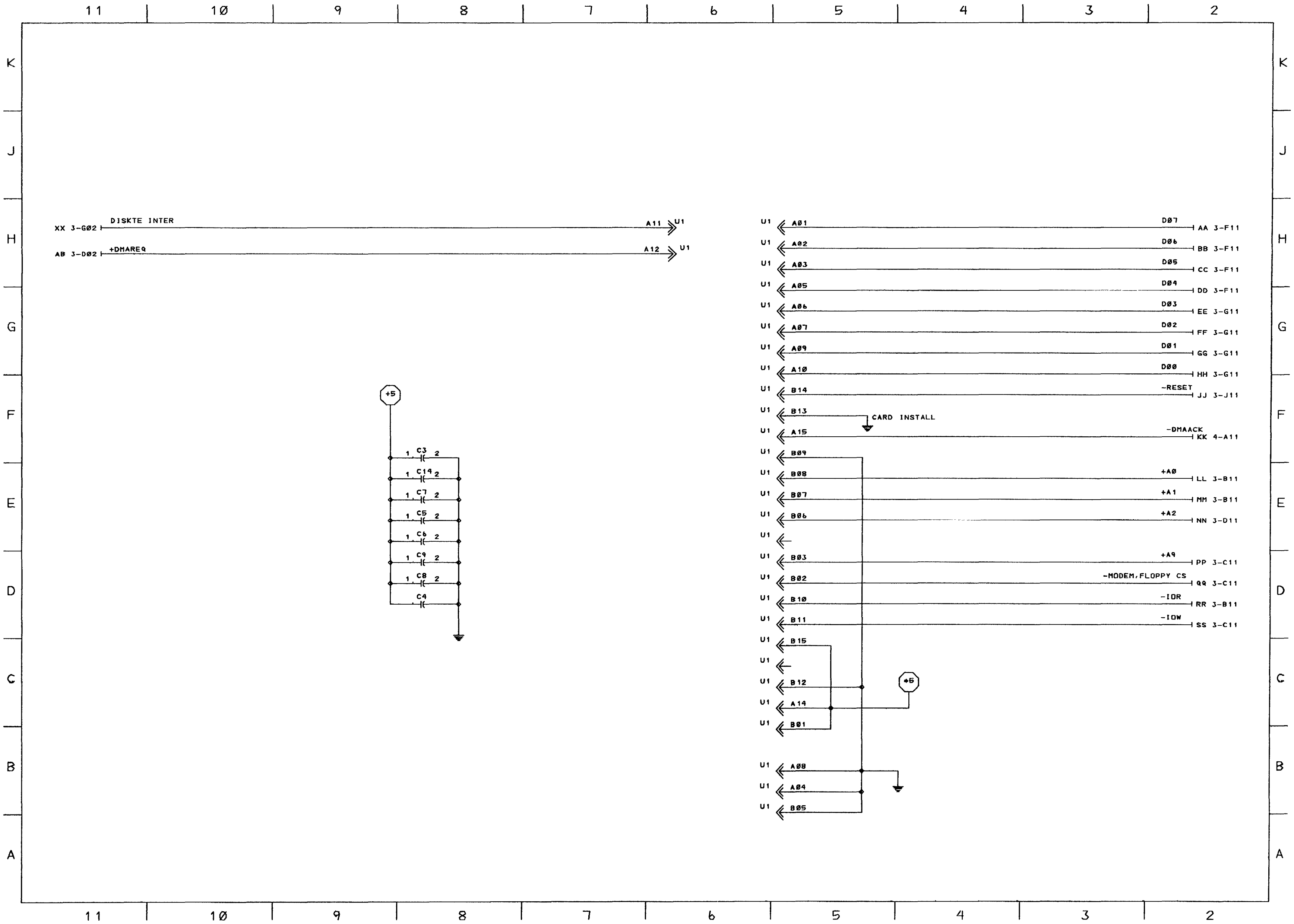
64KB Memory and Display Expansion (Sheet 4 of 4)

DANGER

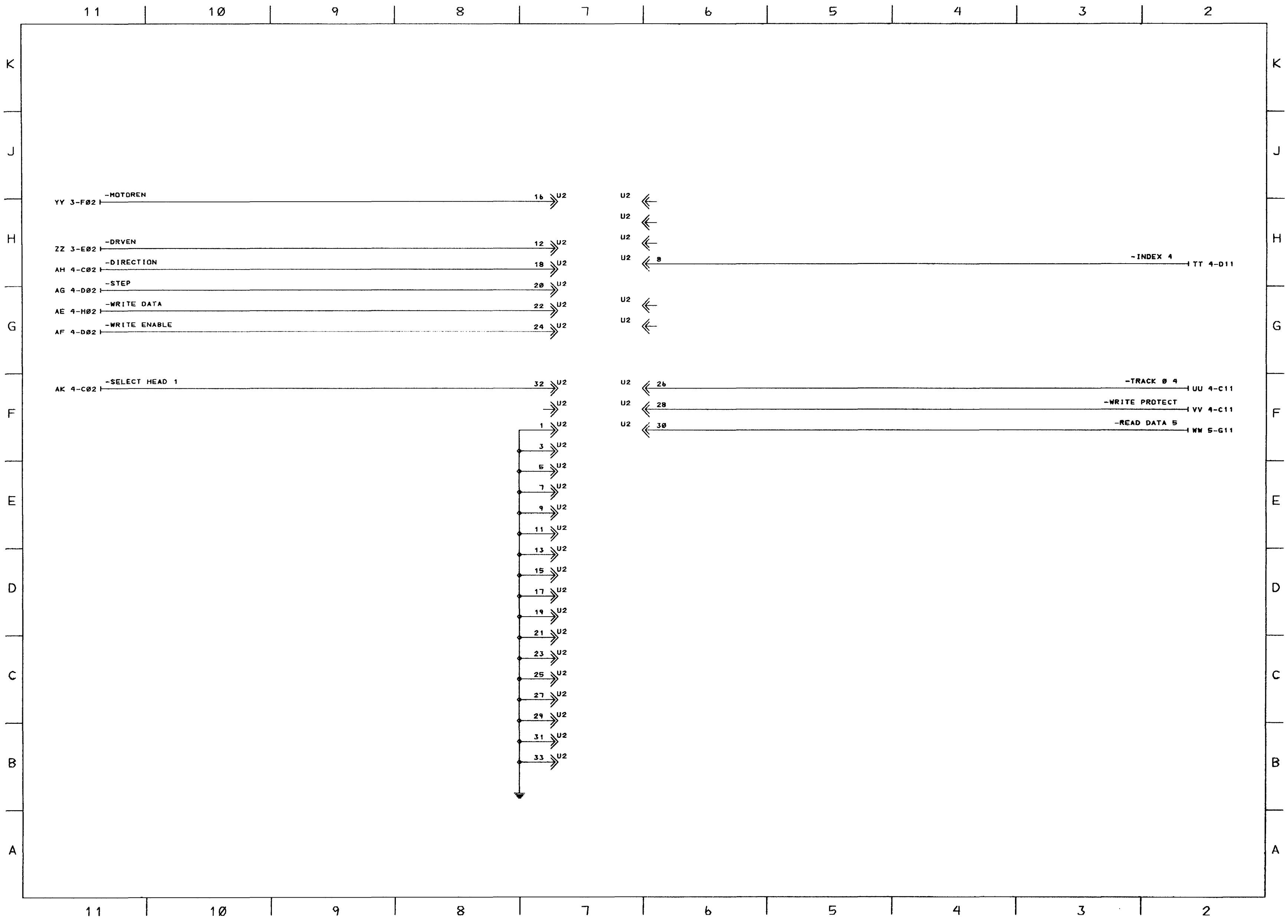
HAZARDOUS VOLTAGES UP TO 450 VOLTS EXIST ON THE PRINTED CIRCUIT BOARDS



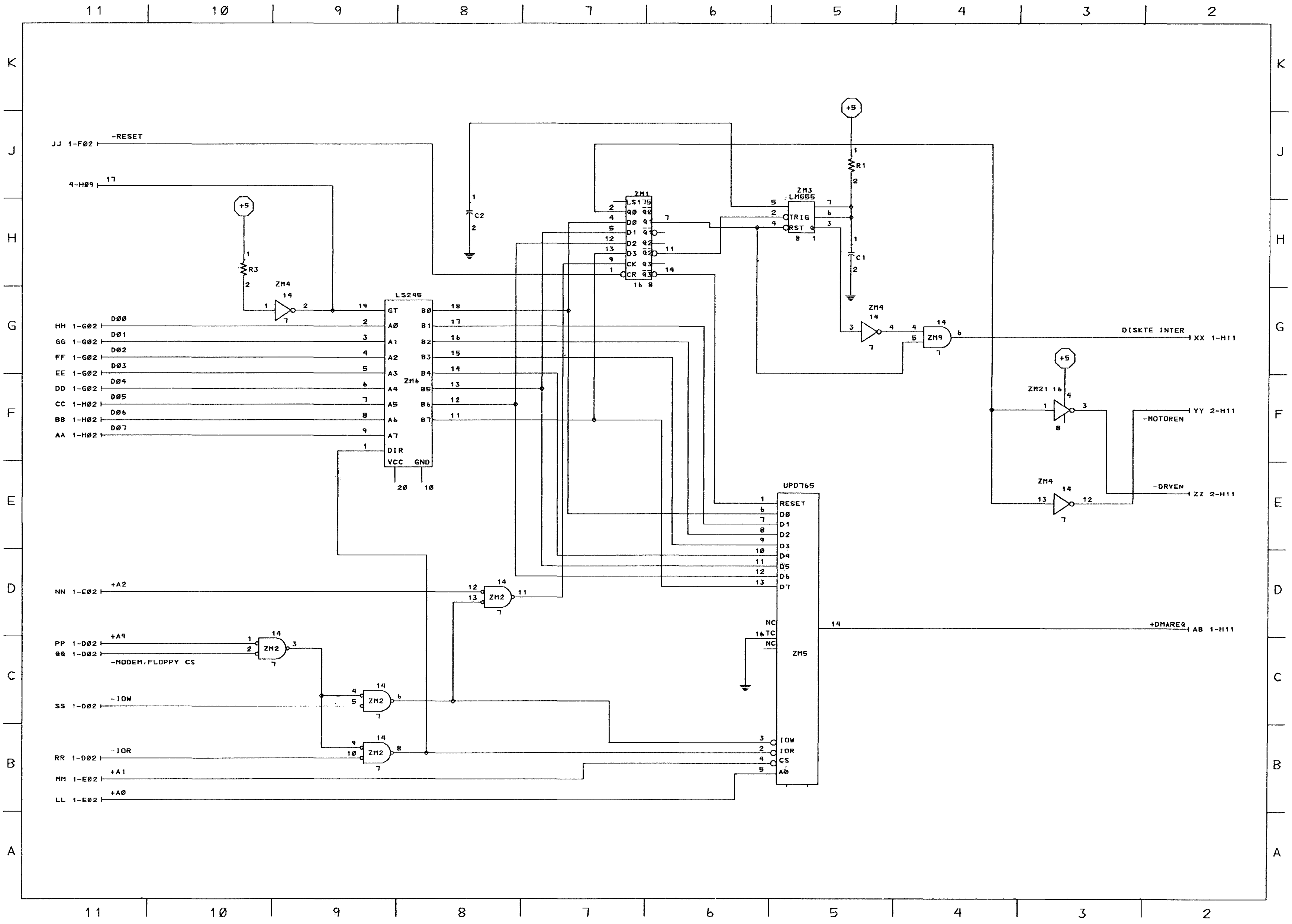
Color Display



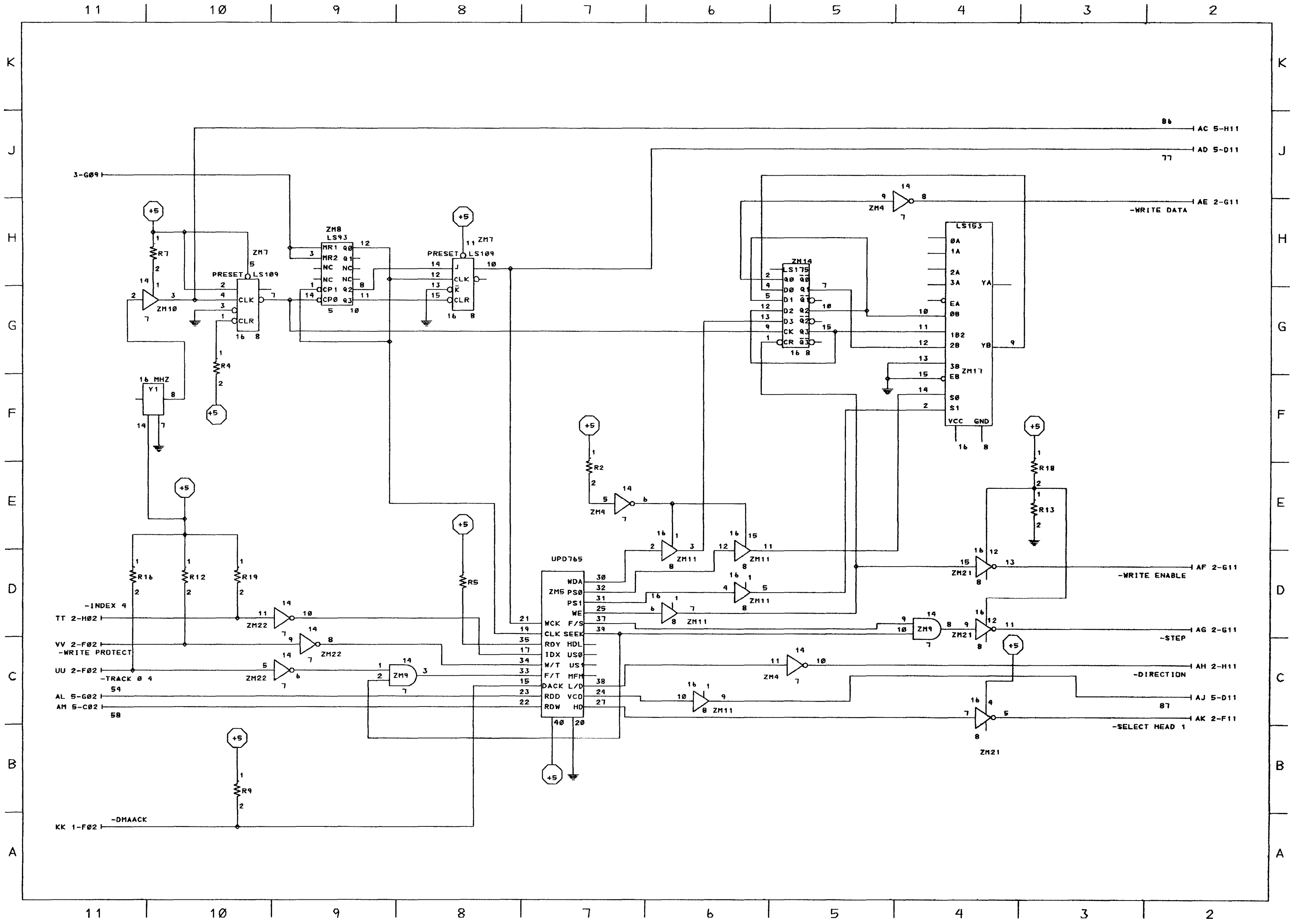
Diskette Drive Adapter (Sheet 1 of 6)



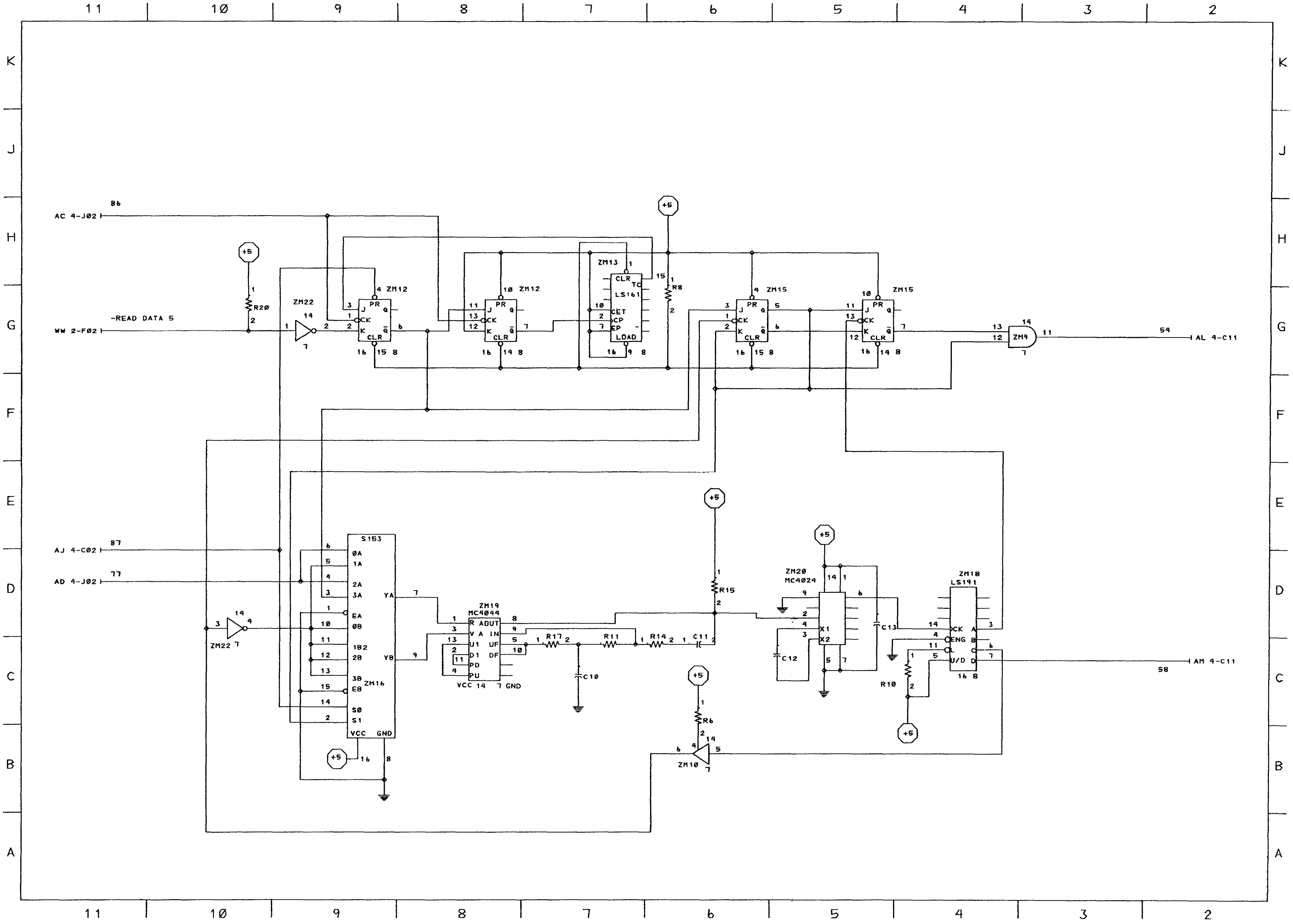
Diskette Drive Adapter (Sheet 2 of 6)



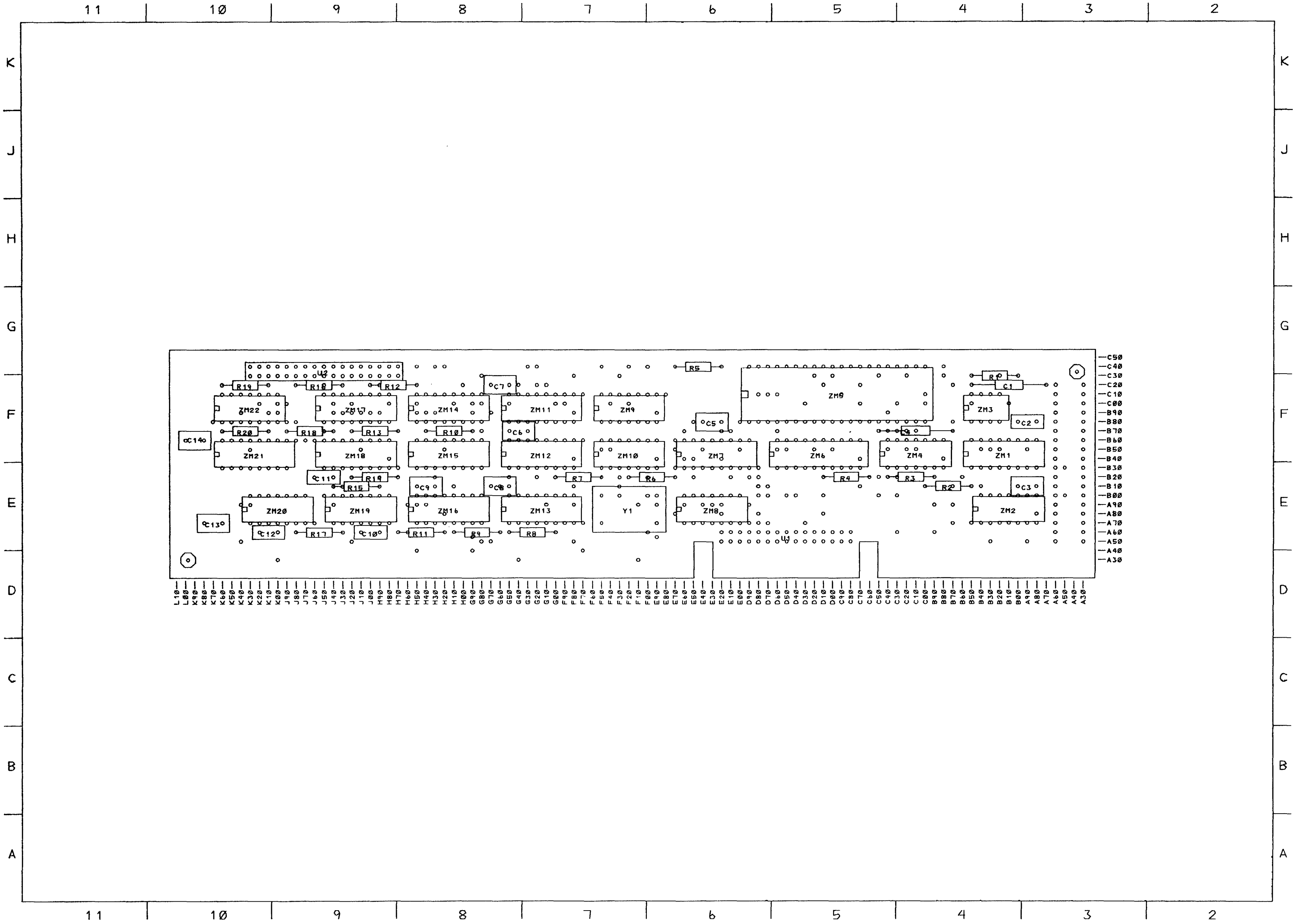
Diskette Drive Adapter (Sheet 3 of 6)



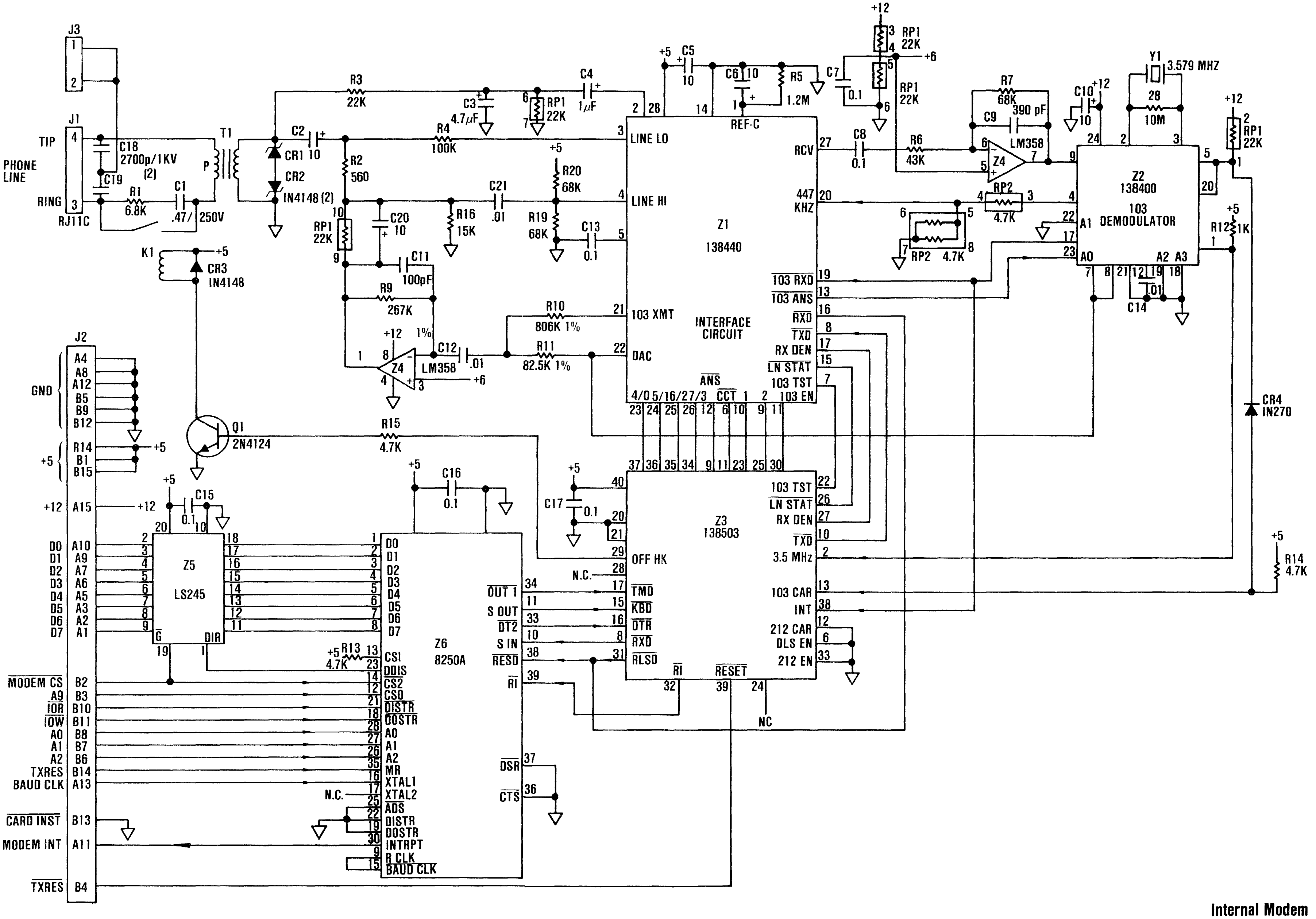
Diskette Drive Adapter (Sheet 4 of 6)



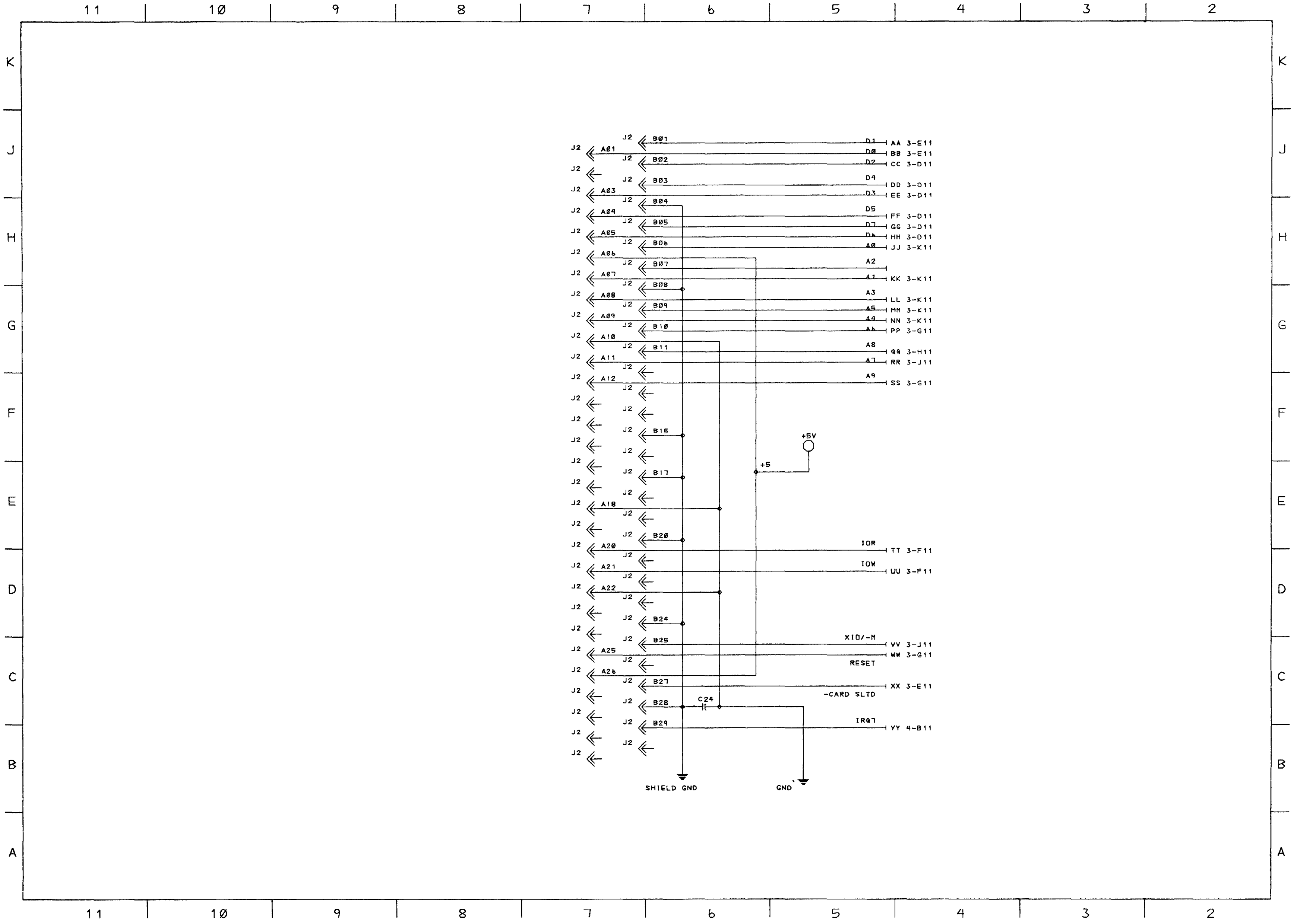
Diskette Drive Adapter (Sheet 5 of 6)



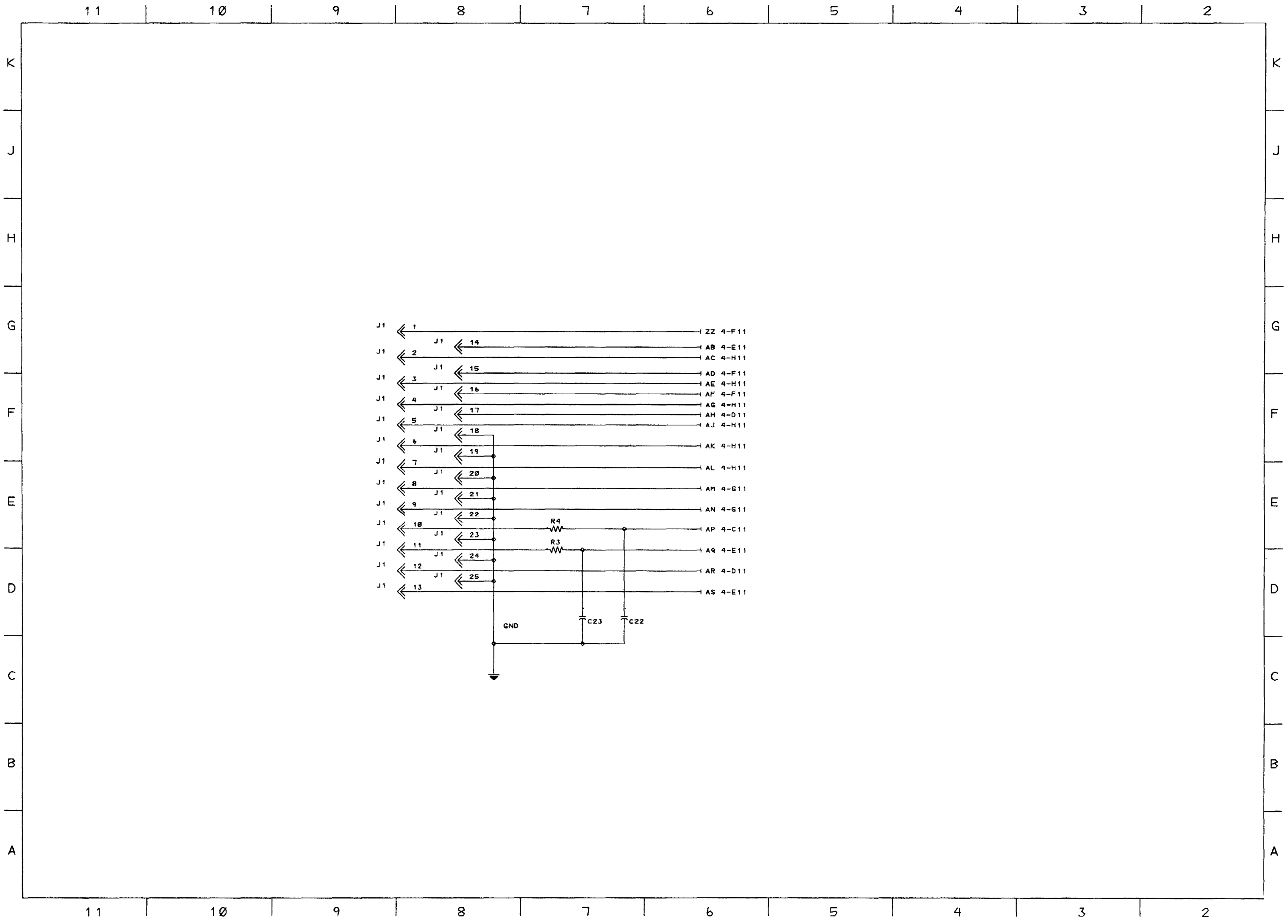
Diskette Drive Adapter (Sheet 6 of 6)



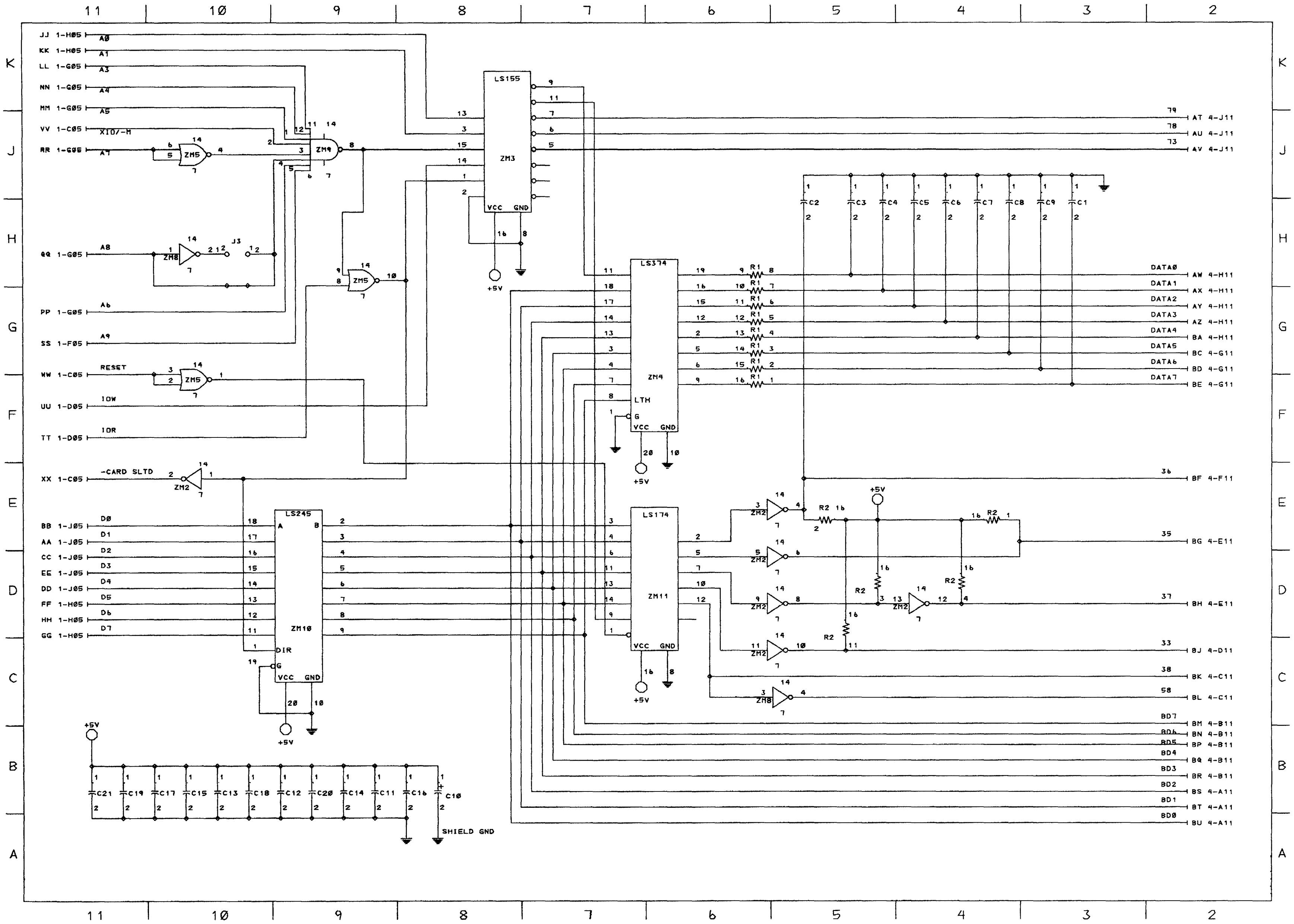
Internal Modem



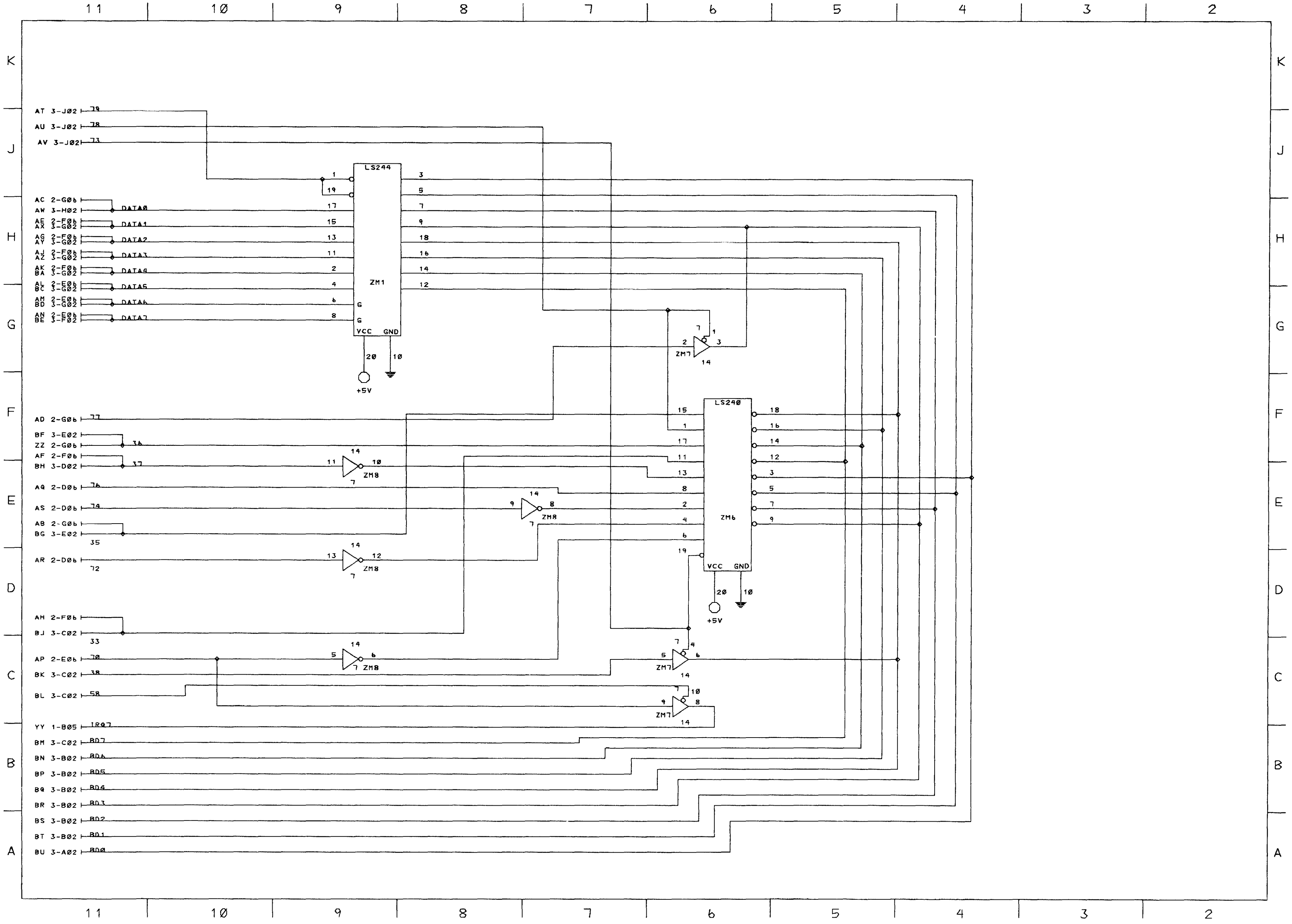
Parallel Printer Attachment (Sheet 1 of 5)



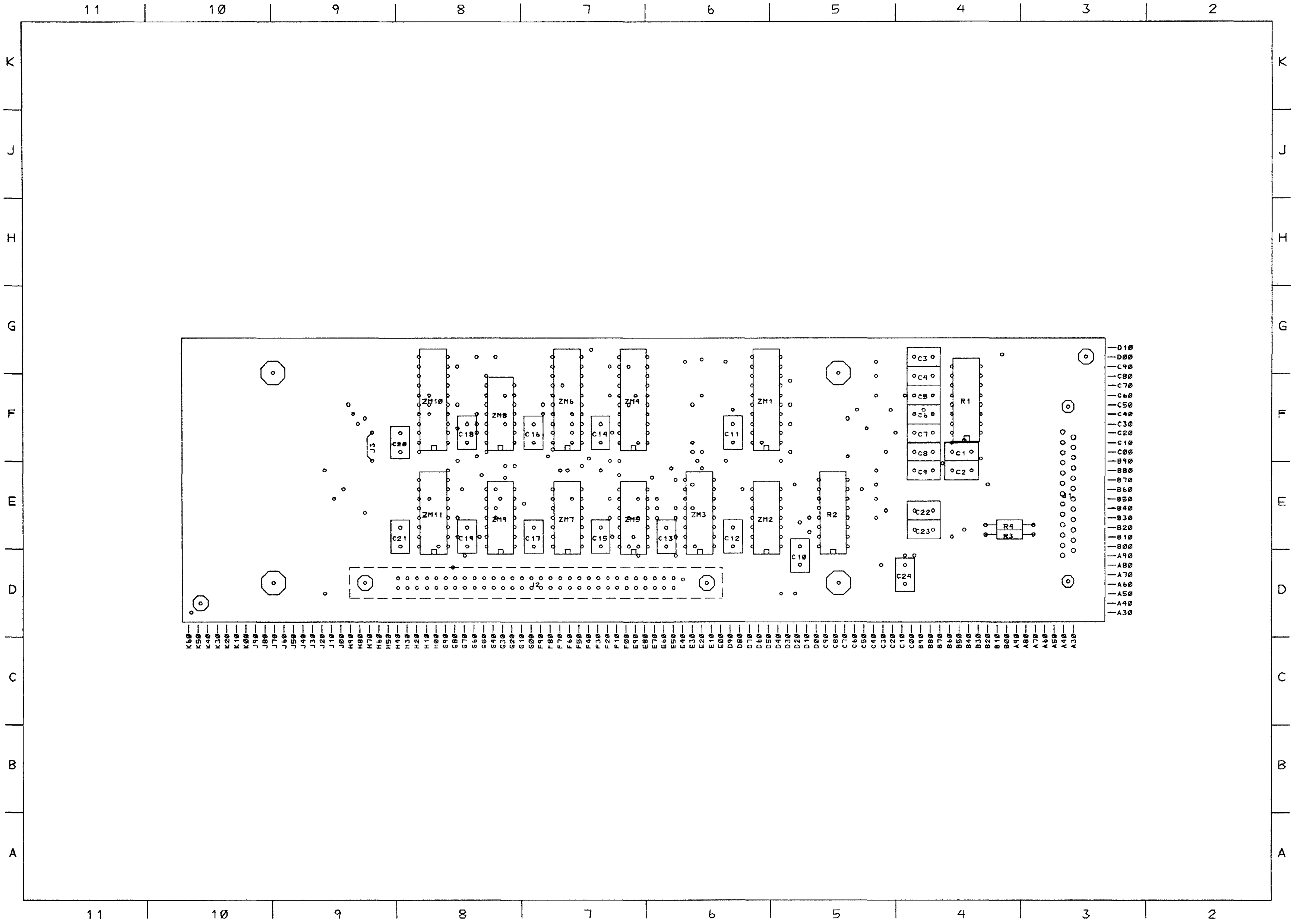
Parallel Printer Attachment (Sheet 2 of 5)



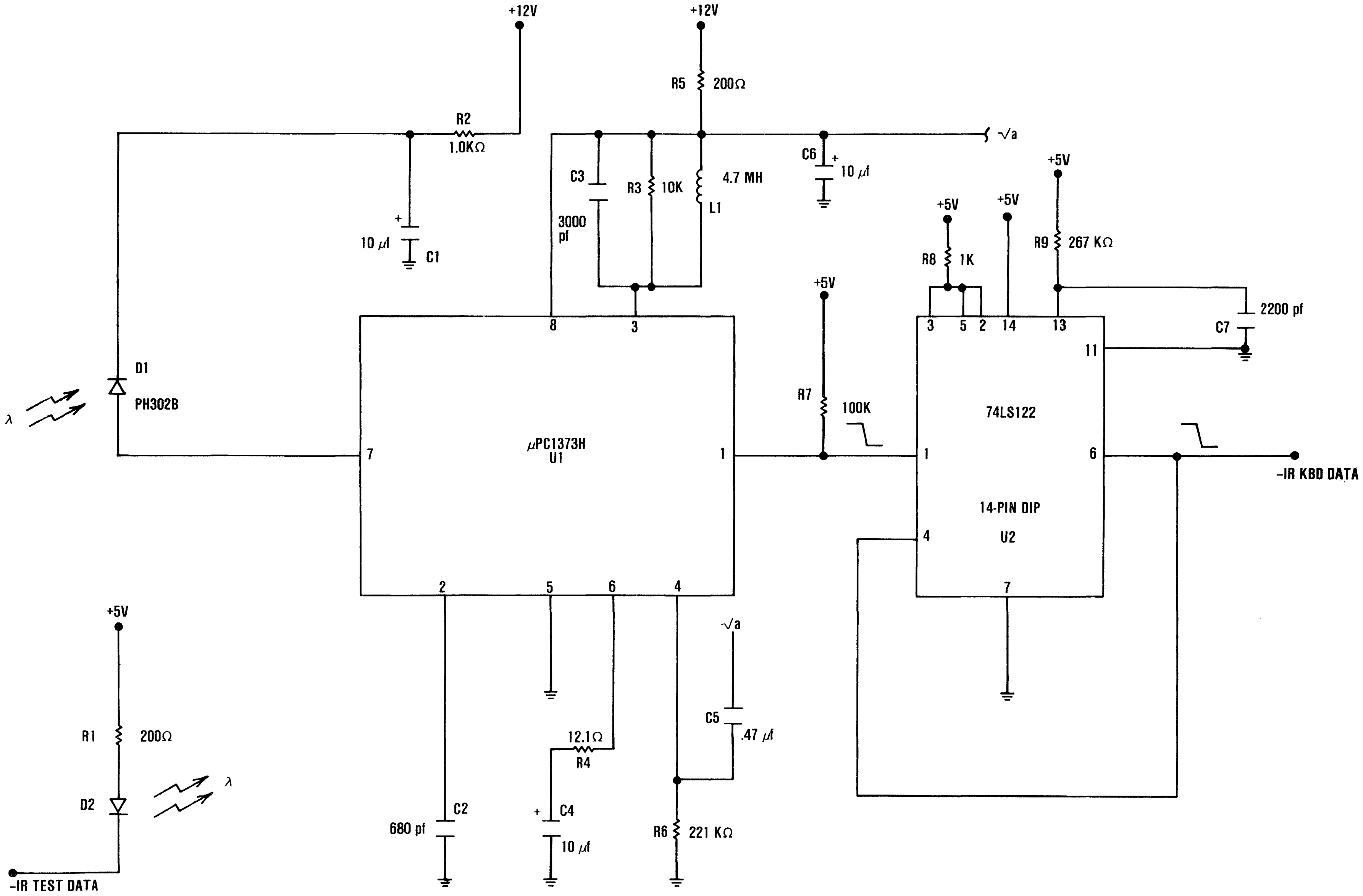
Parallel Printer Attachment (Sheet 3 of 5)



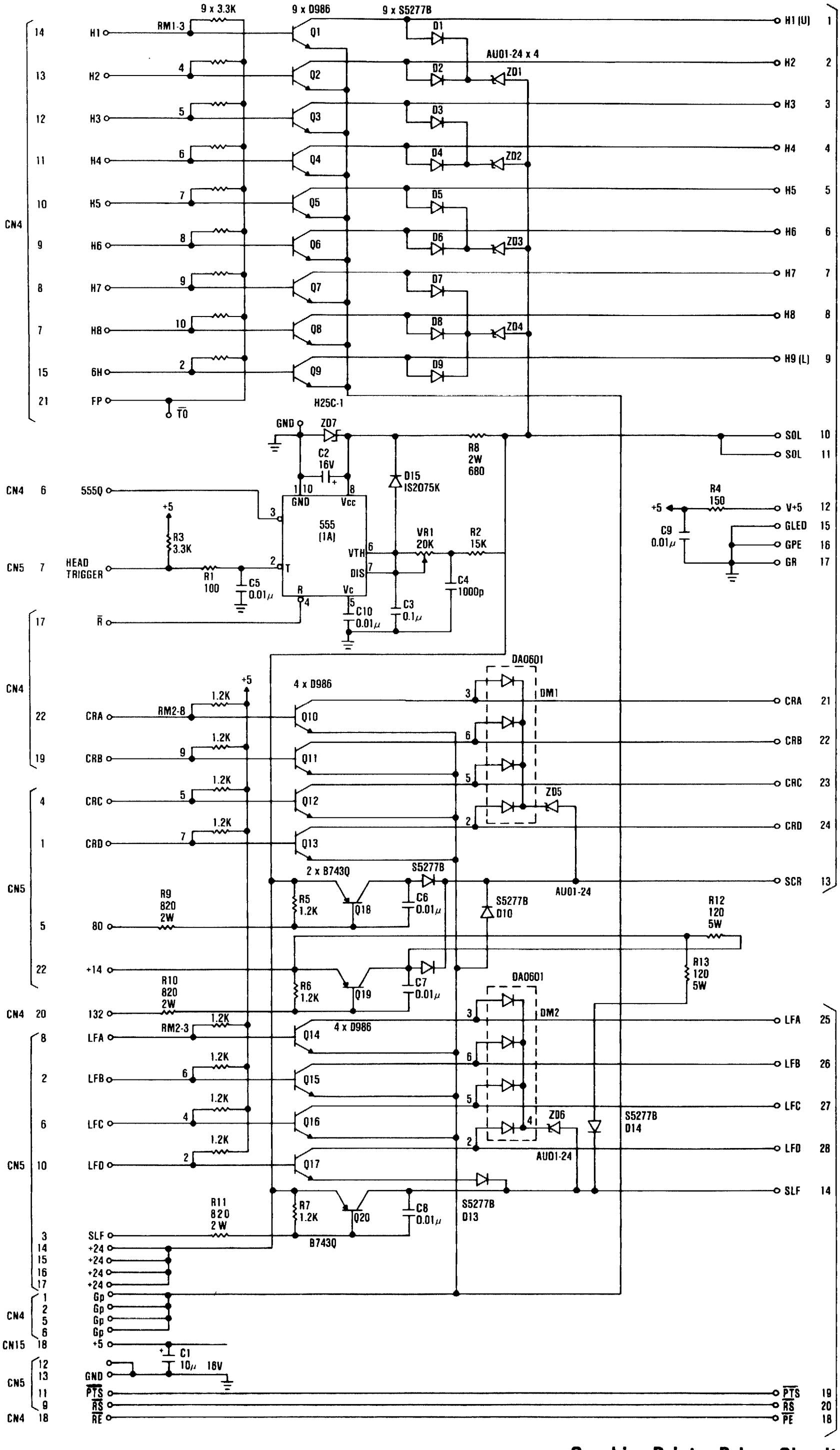
Parallel Printer Attachment (Sheet 4 of 5)



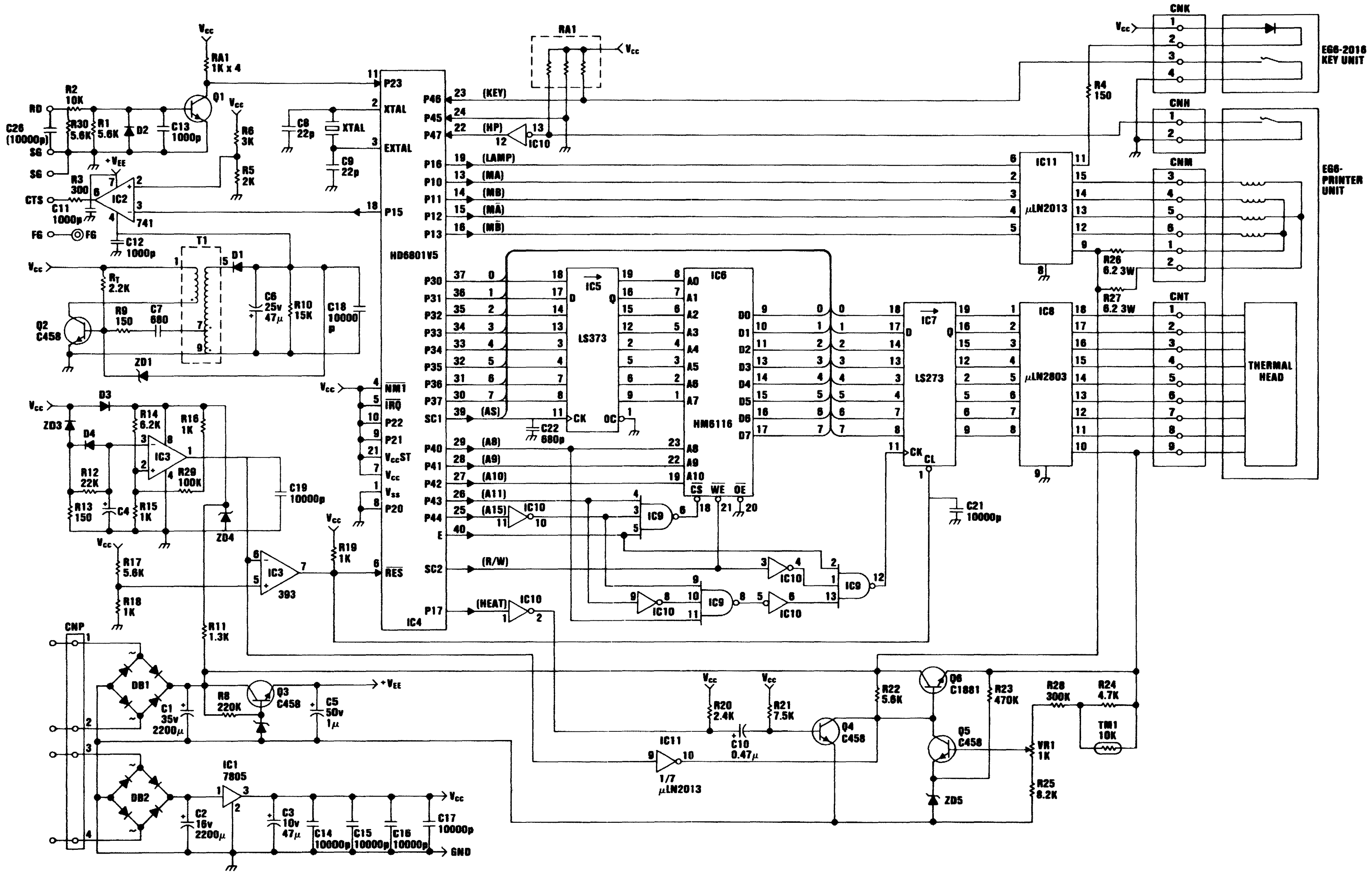
Parallel Printer Attachment (Sheet 5 of 5)



Infra-Red Receiver Board



Graphics Printer Driver Circuit



PC Compact Printer

Bibliography

Intel Corporation. *The 8086 Family User's Manual* This manual introduces the 8086 family of microcomputing components and serves as a reference in system design and implementation.

Intel Corporation. *8086/8087/8088 Macro Assembly Reference Manual for 8088/8085 Based Development System* This manual describes the 8086/8087/8088 Macro Assembly Language, and is intended for use by persons who are familiar with assenbly language.

Intel Corporation. *Component Data Catalog* This book describes Intel components and their technecal specifications.

Motorola, Inc. *The Complete Microcomputer Data Library.* This book describes Motorola components and their technical specifications.

National Semiconductor Corporation. *INS 8250 Asynchronous Communications Element.* This book documents Physical and operating characteristics of the INS 8250.

Notes:

APPENDIX C: CHARACTERS, KEYSTROKES, AND COLOR

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
00	0	Blank (Null)	Ctrl 2		Black	Black
01	1	☺	Ctrl A		Black	Blue
02	2	☻	Ctrl B		Black	Green
03	3	♥	Ctrl C		Black	Cyan
04	4	♦	Ctrl D		Black	Red
05	5	♣	Ctrl E		Black	Magenta
06	6	♠	Ctrl F		Black	Brown
07	7	•	Ctrl G		Black	Light Grey
08	8	◘	Ctrl H, Backspace, Shift Backspace		Black	Dark Grey
09	9	○	Ctrl I		Black	Light Blue
0A	10	◙	Ctrl J, Ctrl ↵		Black	Light Green
0B	11	♂	Ctrl K		Black	Light Green
0C	12	♀	Ctrl L		Black	Light Red
0D	13	♪	Ctrl M, ↵ Shift ↵		Black	Light Magenta
0E	14	♫	Ctrl N		Black	Yellow
0F	15	☼	Ctrl O		Black	White
10	16	►	Ctrl P		Blue	Black
11	17	◄	Ctrl Q		Blue	Blue
12	18	↕	Ctrl R		Blue	Green
13	19	‼	Ctrl S		Blue	Cyan
14	20	¶	Ctrl T		Blue	Red
15	21	§	Ctrl U			Magenta
16	22	▬	Ctrl V		Blue	Brown
17	23	↨	Ctrl W		Blue	Light Grey

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
18	24	↑	Ctrl X		Blue	Dark Grey
19	25	↓	Ctrl Y		Blue	Light Blue
1A	26	→	Ctrl Z		Blue	Light Green
1B	27	←	Ctrl [, Esc, Shift Esc, Ctrl Esc		Blue	Light Cyan
1C	28	∟	Ctrl \		Blue	Light Red
1D	29	↔	Ctrl]		Blue	Light Magenta
1E	30	▲	Ctrl 6		Blue	Yellow
1F	31	▼	Ctrl —		Blue	White
20	32	Blank Space	Space Bar, Shift, Space, Ctrl Space, Alt Space		Green	Black
21	33	!	!	Shift	Green	Blue
22	34	"	"	Shift	Green	Green
23	35	#	#	Shift	Green	Cyan
24	36	$	$	Shift	Green	Red
25	37	%	%	Shift	Green	Magenta
26	38	&	&	Shift	Green	Brown
27	39	'	'		Green	Light Grey
28	40	(	(	Shift	Green	Dark Grey
29	41	)	)	Shift	Green	Light Blue
2A	42	*	*	Note 1	Green	Light Green
28	43	+	+	Shift	Green	Light Cyan
2C	44	,	,		Green	Light Red
2D	45	—	—		Green	Light Magenta
2E	46	.	.	Note 2	Green	Yellow
2F	47	/	/		Green	White
30	48	0	0	Note 3	Cyan	Black
31	49	1	1	Note 3	Cyan	Blue
32	50	2	2	Note 3	Cyan	Green
33	51	3	3	Note 3	Cyan	Cyan

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
34	52	4	4	Note 3	Cyan	Red
35	53	5	5	Note 3	Cyan	Magenta
36	54	6	6	Note 3	Cyan	Brown
37	55	7	7	Note 3	Cyan	Light Grey
38	56	8	8	Note 3	Cyan	Dark Grey
39	57	9	9	Note 3	Cyan	Light Blue
3A	58	:	:	Shift	Cyan	Light Green
3B	59	;	;		Cyan	Light Cyan
3C	60	<	<	Shift	Cyan	Light Red
3D	61	=	=		Cyan	Light Magenta
3E	62	>	>	Shift	Cyan	Yellow
3F	63	?	?	Shift	Cyan	White
40	64	@	@	Shift	Red	Black
41	65	A	A	Note 4	Red	Blue
42	66	B	B	Note 4	Red	Green
43	67	C	C	Note 4	Red	Cyan
44	68	D	D	Note 4	Red	Red
45	69	E	E	Note 4	Red	Magenta
46	70	F	F	Note 4	Red	Brown
47	71	G	G	Note 4	Red	Light Grey
48	72	H	H	Note 4	Red	Dark Grey
49	73	I	I	Note 4	Red	Light Blue
4A	74	J	J	Note 4	Red	Light Green
4B	75	K	K	Note 4	Red	Light Cyan
4C	76	L	L	Note 4	Red	Light Red
4D	77	M	M	Note 4	Red	Light Magenta
4E	78	N	N	Note 4	Red	Yellow
4F	79	O	O	Note 4	Red	White
50	80	P	P	Note 4	Magenta	Black
51	81	Q	Q	Note 4	Magenta	Blue
52	82	R	R	Note 4	Magenta	Green
53	83	S	S	Note 4	Magenta	Cyan
54	84	T	T	Note 4	Magenta	Red

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
55	85	U	U	Note 4	Magenta	Magenta
56	86	V	V	Note 4	Magenta	Brown
57	57	W	W	Note 4	Magenta	Light Grey
58	88	X	X	Note 4	Magenta	Dark Grey
59	89	Y	Y	Note 4	Magenta	Light Blue
5A	90	Z	Z	Note 4	Magenta	Light Green
5B	91	[	[		Magenta	Light Cyan
5C	92	\	\		Magenta	Light Red
5D	93	]	]		Magenta	Light Magenta
5E	94	^	^	Shift	Magenta	Yellow
5F	95	—	—	Shift	Magenta	White
60	96	`	`		Yellow	Black
61	97	a	a	Note 5	Yellow	Blue
62	98	b	b	Note 5	Yellow	Green
63	99	c	c	Note 5	Yellow	Cyan
64	100	d	d	Note 5	Yellow	Red
65	101	e	e	Note 5	Yellow	Magenta
66	102	f	f	Note 5	Yellow	Brown
67	103	g	g	Note 5	Yellow	Light Grey
68	104	h	h	Note 5	Yellow	Dark Grey
69	105	i	i	Note 5	Yellow	Light Blue
6A	106	j	j	Note 5	Yellow	Light Green
6B	107	k	k	Note 5	Yellow	Light Cyan
6C	108	l	l	Note 5	Yellow	Light Red
6D	109	m	m	Note 5	Yellow	Light Magenta
6E	110	n	n	Note 5	Yellow	Yellow
6F	111	o	o	Note 5	Yellow	White
70	112	p	p	Note 5	White	Black
71	113	q	q	Note 5	White	Blue
72	114	r	r	Note 5	White	Green
73	115	s	s	Note 5	White	Cyan
74	116	f	f	Note 5	White	Red
75	117	u	u	Note 5	White	Magenta
76	118	v	v	Note 5	White	Brown

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
77	119	w	w	Note 5	White	Light Grey
78	120	x	x	Note 5	White	Dark Grey
79	121	y	y	Note 5	White	Light Blue
7A	122	z	z	Note 5	White	Light Green
7B	123	{	{	Shift	White	Light Cyan
7C	124	¦	¦	Shift	White	Light Red
7D	125	}	}	Shift	White	Light Magenta
7E	126	~	~	Shift	White	Yellow
7F	127	Δ	Ctrl ←		White	White
* * * *		80 to FF Hex are Flashing if Blink is Enabled				* * * *
80	128	Ç	Alt 128	Note 6	Black	Black
81	129	ü	Alt 129	Note 6	Black	Blue
82	130	é	Alt 130	Note 6	Black	Green
83	131	â	Alt 131	Note 6	Black	Cyan
84	132	ä	Alt 132	Note 6	Black	Red
85	133	à	Alt 133	Note 6	Black	Magenta
86	134	å	Alt 134	Note 6	Black	Brown
87	135	ç	Alt 135	Note 6	Black	Light Grey
88	136	ê	Alt 136	Note 6	Black	Dark Grey
89	137	ë	Alt 137	Note 6	Black	Light Blue
8A	138	è	Alt 138	Note 6	Black	Light Green
8B	139	ï	Alt 139	Note 6	Black	Light Cyan
8C	140	î	Alt 140	Note 6	Black	Light Red
8D	141	ì	Alt 141	Note 6	Black	Light Magenta
8E	142	Ä	Alt 142	Note 6	Black	Yellow
8F	143	Å	Alt 143	Note 6	Black	White
90	144	É	Alt 144	Note 6	Blue	Black
91	145	æ	Alt 145	Note 6	Blue	Blue
92	146	Æ	Alt 146	Note 6	Blue	Green
93	147	ô	Alt 147	Note 6	Blue	Cyan
94	148	ö	Alt 148	Note 6	Blue	Red
95	149	ò	Alt 149	Note 6	Blue	Magenta

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
96	150	û	Alt 150	Note 6	Blue	Brown
97	151	ù	Alt 151	Note 6	Blue	Light Grey
98	152	ÿ	Alt 152	Note 6	Blue	Dark Grey
99	153	ó	Alt 153	Note 6	Blue	Light Blue
9A	154	ü	Alt 154	Note 6	Blue	Light Green
9B	155	¢	Alt 155	Note 6	Blue	Light Cyan
9C	156	£	Alt 156	Note 6	Blue	Light Red
9D	157	¥	Alt 157	Note 6	Blue	Light Magenta
9E	158	Pt	Alt 158	Note 6	Blue	Yellow
9F	159	ƒ	Alt 159	Note 6	Blue	White
A0	160	á	Alt 160	Note 6	Green	Black
A1	161	í	Alt 161	Note 6	Green	Blue
A2	162	ó	Alt 162	Note 6	Green	Green
A3	163	ú	Alt 163	Note 6	Green	Cyan
A4	164	ñ	Alt 164	Note 6	Green	Red
A5	165	Ñ	Alt 165	Note 6	Green	Magenta
A6	166	ª	Alt 166	Note 6	Green	Brown
A7	167	º	Alt 167	Note 6	Green	Light Grey
A8	168	¿	Alt 168	Note 6	Green	Dark Grey
A9	169	⌐	Alt 169	Note 6	Green	Light Blue
AA	170	¬	Alt 170	Note 6	Green	Light Green
AB	171	½	Alt 171	Note 6	Green	Light Cyan
AC	172	¼	Alt 172	Note 6	Green	Light Red
AD	173	¡	Alt 173	Note 6	Green	Light Magenta
AE	174	«	Alt 174	Note 6	Green	Yellow
AF	175	»	Alt 175	Note 6	Green	White
B0	176	░	Alt 176	Note 6	Cyan	Black
B1	177	▒	Alt 177	Note 6	Cyan	Blue
B2	178	▓	Alt 178	Note 6	Cyan	Green
B3	179	│	Alt 179	Note 6	Cyan	Cyan
B4	180	┤	Alt 180	Note 6	Cyan	Red
B5	181	╡	Alt 181	Note 6	Cyan	Magenta
B6	182	╢	Alt 182	Note 6	Cyan	Brown

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
B7	183	╖	Alt 183	Note 6	Cyan	Light Grey
B8	184	╕	Alt 184	Note 6	Cyan	Dark Grey
B9	185	╣	Alt 185	Note 6	Cyan	Light Blue
BA	186	║	Alt 186	Note 6	Cyan	Light Green
BB	187	╗	Alt 187	Note 6	Cyan	Light Cyan
BC	188	╝	Alt 188	Note 6	Cyan	Light Red
BD	189	╜	Alt 189	Note 6	Cyan	Light Magenta
BE	190	╛	Alt 190	Note 6	Cyan	Yellow
BF	191	┐	Alt 191	Note 6	Cyan	White
C0	192	└	Alt 192	Note 6	Red	Black
C1	193	┴	Alt 193	Note 6	Red	Blue
C2	194	┬	Alt 194	Note 6	Red	Green
C3	195	├	Alt 195	Note 6	Red	Cyan
C4	196	─	Alt 196	Note 6	Red	Red
C5	197	┼	Alt 197	Note 6	Red	Magenta
C6	198	╞	Alt 198	Note 6	Red	Brown
C7	199	╟	Alt 199	Note 6	Red	Light Grey
C8	200	╚	Alt 200	Note 6	Red	Dark Grey
C9	201	╔	Alt 201	Note 6	Red	Light Blue
CA	202	╩	Alt 202	Note 6	Red	Light Green
CB	203	╦	Alt 203	Note 6	Red	Light Cyan
CC	204	╠	Alt 204	Note 6	Red	Light Red
CD	205	═	Alt 205	Note 6	Red	Light Magenta
CE	206	╬	Alt 206	Note 6	Red	Yellow
CF	207	╧	Alt 207	Note 6	Red	White
D0	208	╨	Alt 208	Note 6	Magenta	Black
D1	209	╤	Alt 209	Note 6	Magenta	Blue
D2	210	╥	Alt 210	Note 6	Magenta	Green
D3	211	╙	Alt 211	Note 6	Magenta	Cyan
D4	212	╘	Alt 212	Note 6	Magenta	Red
D5	213	╒	Alt 213	Note 6	Magenta	Magenta
D6	214	╓	Alt 214	Note 6	Magenta	Brown
D7	215	╫	Alt 215	Note 6	Magenta	Light Grey

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
D8	216	╪	Alt 216	Note 6	Magenta	Dark Grey
D9	217	┘	Alt 217	Note 6	Magenta	Light Blue
DA	218	┌	Alt 218	Note 6	Magenta	Light Green
DB	219	█	Alt 219	Note 6	Magenta	Light Cyan
DC	220	▄	Alt 220	Note 6	Magenta	Light Red
DD	221	▌	Alt 221	Note 6	Magenta	Light Magenta
DE	222	▐	Alt 222	Note 6	Magenta	Yellow
DF	223	▀	Alt 223	Note 6	Magenta	White
E0	224	α	Alt 224	Note 6	Yellow	Black
E1	225	β	Alt 225	Note 6	Yellow	Blue
E2	226	Γ	Alt 226	Note 6	Yellow	Green
E3	227	π	Alt 227	Note 6	Yellow	Cyan
E4	228	Σ	Alt 228	Note 6	Yellow	Red
E5	229	σ	Alt 229	Note 6	Yellow	Magenta
E6	230	μ	Alt 230	Note 6	Yellow	Brown
E7	231	τ	Alt 231	Note 6	Yellow	Light Grey
E8	232	Φ	Alt 232	Note 6	Yellow	Dark Grey
E9	233	θ	Alt 233	Note 6	Yellow	Light Blue
EA	234	Ω	Alt 234	Note 6	Yellow	Light Green
EB	235	δ	Alt 235	Note 6	Yellow	Light Cyan
EC	236	∞	Alt 236	Note 6	Yellow	Light Red
ED	237	φ	Alt 237	Note 6	Yellow	Light Magenta
EE	238	ϵ	Alt 238	Note 6	Yellow	Yellow
EF	239	∩	Alt 239	Note 6	Yellow	White
F0	240	≡	Alt 240	Note 6	White	Black
F1	241	±	Alt 241	Note 6	White	Blue
F2	242	≥	Alt 242	Note 6	White	Green
F3	243	≤	Alt 243	Note 6	White	Cyan
F4	244	⌠	Alt 244	Note 6	White	Red
F5	245	⌡	Alt 245	Note 6	White	Magenta
F6	246	÷	Alt 246	Note 6	White	Brown
F7	247	≈	Alt 247	Note 6	White	Light Grey
F8	248	°	Alt 248	Note 6	White	Dark Grey

Value		As Characters			Color/Graphics Text Attributes	
Hex	Dec	Symbol	Keystrokes	Modes	Background	Foreground
F9	249	●	Alt 249	Note 6	White	Light Blue
FA	250	•	Alt 250	Note 6	White	Light Green
FB	251	√	Alt 251	Note 6	White	Light Cyan
FC	252	η	Alt 252	Note 6	White	Light Red
FD	253	2	Alt 253	Note 6	White	Light Magenta
FE	254	■	Alt 254	Note 6	White	Yellow
FF	255	BLANK	Alt 255	Note 6	White	White

NOTE 1 On the 62-key keyboard the Asterisk (*) can be keyed using two methods: 1) in the shift mode hit the * 8 key or 2) hold Alt key and press the < , key.

On the 83-key keyboard the Asterisk (*) can be keyed using two methods: 1) hit the Prt Sc * key or 2) in the shift mode hit the * 8 key.

NOTE 2 Period (.) can easily be keyed using two methods: 1) hit the > • key or 2) in shift or Num Lock mode hit the • Del key.

NOTE 3 Numeric characters (0—9) can easily be keyed using two methods: 1) hit the numeric keys on the top row of the typewriter portion of the keyboard or 2) on the 83-key keyboard in shift or Num Lock mode hit the numeric keys in the 10—key pad portion of the keyboard.

NOTE 4 Upper case alphabetic characters (A—Z) can easily be keyed in two modes: 1) in shift mode the appropriate alphabetic key or 2) In Caps Lock mode hit the appropriate alphabetic key.

NOTE 5 Lower case alphabetic characters (a—z) can easily be keyed in two modes: 1) in "normal" mode hit the appropriate key or 2) In Caps Lock combined with shift mode hit the appropriate alphabetic key.

NOTE 6 On the 62-key keyboard set Num Lock state using Alt/Fn/N then 3 digits after the Alt key must be typed from the numeric keys on the top row of the typematic portion of the keyboard. Character codes 000 through 255 can be entered in this fashion. (With Caps Lock activated, character codes 97 through 122 will display upper case rather than lower case alphabetic characters.)

On the 83-key keyboard the 3 digits after the Alt key must be typed from the numeric key pad (keys 71—73, 75—77, 79—82).

Character Set (00-7F) Quick Reference

DECIMAL VALUE	➧	0	16	32	48	64	80	96	112
⬇	HEXA-DECIMAL VALUE	0	1	2	3	4	5	6	7
0	0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	`	p
1	1	☺	◄	!	1	A	Q	a	q
2	2	☻	↕	"	2	B	R	b	r
3	3	♥	‼	#	3	C	S	c	s
4	4	♦	¶	$	4	D	T	d	t
5	5	♣	§	%	5	E	U	e	u
6	6	♠	▬	&	6	F	V	f	v
7	7	•	↨	'	7	G	W	g	w
8	8	◘	↑	(	8	H	X	h	x
9	9	○	↓	)	9	I	Y	i	y
10	A	◙	→	*	:	J	Z	j	z
11	B	♂	←	+	;	K	[	k	{
12	C	♀	∟	,	<	L	\	l	¦
13	D	♪	↔	—	=	M	]	m	}
14	E	♫	▲	.	>	N	^	n	~
15	F	☼	▼	/	?	O	_	o	Δ

Character Set (80-FF) Quick Reference

DECIMAL VALUE	➡	128	144	160	176	192	208	224	240
⬇	HEXA-DECIMAL VALUE	8	9	A	B	C	D	E	F
0	0	Ç	É	á	░	└	╨	∝	≡
1	1	ü	æ	í	▒	┴	╤	β	±
2	2	é	Æ	ó	▓	┬	╥	Γ	≥
3	3	â	ô	ú	│	├	╙	π	≤
4	4	ä	ö	ñ	┤	─	╘	Σ	⌠
5	5	à	ò	Ñ	╡	┼	╒	σ	⌡
6	6	å	û	ª	╢	╞	╓	μ	÷
7	7	ç	ù	º	╖	╟	╫	τ	≈
8	8	ê	ÿ	¿	╕	╚	╪	Φ	°
9	9	ë	Ö	⌐	╣	╔	┘	Θ	●
10	A	è	Ü	¬	║	╩	┌	Ω	•
11	B	ï	¢	½	╗	╦	█	δ	√
12	C	î	£	¼	╝	╠	▄	∞	n
13	D	ì	¥	¡	╜	═	▌	φ	2
14	E	Ä	₧	«	╛	╬	▐	∈	▮
15	F	Å	ƒ	»	┐	╧	▀	∩	BLANK 'FF'

Appendix D. UNIT SPECIFICATIONS

System Unit

Size:

Length	354 mm (13.9 in.)
Depth	290 mm (11.4 in.)
Height	97 mm (3.8 in.)

Weight:

3.71 Kg (8lb 4oz)	With Diskette Drive
2.61 Kg (5lb 8oz)	Without Diskette Drive

Transformer:

Electrical:

Input	110 Vac 60 Hz
Output to System	Pin 1 - 17 Vac, Pin 2 - GND, Pin 3 - 17 Vac

Power Cords:

Input Length	1.86 meters (6.14 feet)
Type	18 AWG
Output Length	1.22 meters (4.02 feet)
Type	18 AWG

Environment:

Air Temperature	
System ON	**15.6 to 32.2 degrees C (60 to 90 degrees F)**
System Off	**10 to 43 degrees C (50 to 110 degrees F)**
Humidity	
System On	**8% to 80%**
System Off	**8% to 80%**
Noise Level	45 dB

Cordless Keyboard

Size:

Length	341.5 mm (13.45 in.)
Depth	168 mm (6.61 in.)
Height	26 mm (1.02 in.)

Weight:

With Batteries	616 grams (22 ounces)
Without Batteries	700 grams (25 ounces)

Optional Cable:

6 feet, flat

Diskette Drive

Size:

Height	41.6 mm (1.6 in.)
Depth	146 mm (5.8 in.)
Width	208 mm (8.3 in.)

Weight:

1.1 kilograms (2.2 pounds)

Diskette Drive

Power:

Supply

Voltage	**+5 Vdc Input**	**+12 Vdc Input**
Nominal	+5 Vdc	+12 Vdc

Ripple

	+5 Vdc Input	**+12 Vdc Input**
0 to 50 kHz	100 mV	100 mV

Tolerance

	+5 Vdc Input	**+12 Vdc Input**
Including Ripple	+/- 5%	+/- 5%

Standby Current

	+5 Vdc Input	+12 Vdc Input
Nominal	600 mA	400 mA
Worst Case	700 mA	500 mA

Operating Current

	+5 Vdc Input	+12 Vdc Input
Nominal	600 mA	900 mA
Worst Case	700 mA	2400 mA

Mechanical and Electrical

Media	Industry-compatible 5 1/4 inch diskette
Media Life (Head Loaded)	3,000,000 revolutions/track
Media Life (Insertions)	30,000
Tracks Density	48 tracks/inch
Number of Tracks	40
Motor Start Time	500 ms
Instantaneous Speed Variation	+/- 3.0%
Rotational Speed	300 rpm +/- 1.5% (long term)
Nominal Transfer Rate (MFM)	250,000 pulses/second
MTBF (25% Operating)	8,000 POH
Read Bit Shift	+/- 800 ns maximum
Seek Time	6 ms track-to-track maximum
Head Life	20,000 hours (normal use)
Head Load Time	Not Applicable
Head Settling Time	21 ms maximum (from last step pulse)
Error Rate	

	Soft Error	1 per 1,000,000,000 bits maximum (recoverable within 10 retries)
	Hard Error	1 per 1,000,000,000,000 bits maximum (nonrecoverable within 10 retries)
	Access Error	1 per 3,000,000 seeks maximum
Temperature (Exclusive of media)		
	Operating	50 to 122 degrees F (10 to 44 degrees C)
	Non-operating	-40 to 140 degrees F (-40 to 60 degrees C)
Relative Humidity (Exclusive of media)		
	Operating	20 to 80% (noncondensing)
	Non-operating	5 to 95% (noncondensing)
Operating Altitude	7,000 feet above sea level	
Operating Vibration	5 to 500 Hz 11G	

Color Display

Size:

Height	297 mm (11.7 in.)
Depth	407 mm (15.6 in.)
Width	392 mm (15.4 in.)

Weight:

11.8 kilograms (26 pounds)

Heat Output:

240 BTU/hour

Power Cables:

Length	1.83 meters (6 feet)
Size	22 AWG

Graphics Printer

Size:

Height	110 mm (4.3 in.)
Depth	370 mm (14.5 in.)
Width	400 mm (15.7 in.)

Weight:

5.9 kilograms (12.9 pounds)

Heat Output:

341 BTU/hour

Power Cable:

Length	1.83 meters (6 feet)
Size	18 AWG

Signal Cable:

Length	1.83 meters (6 feet)
Size	22 AWG

Electrical:

Minimum	104 Vac
Nominal	120 Vac
Maximum	127 Vac

Internal Modem

Power:

Parameter	**+ 5 Vdc Voltage**	**+ 12 Vdc Voltage**
Tolerance	+/- 5%	+/- 10%
Ripple	50 mV, P-P	50 mV, P-P
Maximum Current	300 mA	50 mA
Current Nominal	150 mA	25 mA

Interface

RS232C

Compact Printer

Size:

Height	88.9 mm (3.5 in)
Depth	221 mm (8.7 in)
Width	312.4 mm (12.3 in)

Weight:

2.99 kg (6.6 lb)

Heat Output:

54.6 Btu/hr

Power Cable:

Length	1.89 mm (6 ft)
Size	28 AWG

Signal Cable:

Length	1.89 m (6 ft)
Size	3 by 18 AWG

Electrical:

Voltage	110 Vac 60 Hz

Glossary

*μ*s Microsecond.

adapter. An auxiliary system or unit used to extend the operation of another system.

address bus. One or more conductors used to carry the binary-coded address from the microprocessor throughout the rest of the system.

all points addressable (APA). A mode in which all points on a displayable image can be controlled by the user.

alphanumeric (A/N). Pertaining to a character set that contains letters, digits, and usually other characters, such as punctuation marks. Synonymous with alphameric.

American Standard Code for Information Interchange. (ASCII) The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), used for information interchange among data processing systems, data communication systems and associated equipment. The ASCII set consists of control characters and graphic characters.

A/N. Alphanumeric.

analog. (1) pertaining to data in the form of continuously variable physical quantities. (2) Contrast with digital.

AND. A logic operator having the property that if P is a statement, Q is a statement, R is a statement,..., then the AND of P, Q, R,...is true if all statements are true, false if any statement is false.

APA. All points addressable.

ASCII. American Standard Code for Information Interchange.

assembler. A computer program used to assemble. Synonymous with assembly program.

asynchronous communications. A communication mode in which each single byte of data is synchronized, usually by the addition of start/stop bits.

BASIC. Beginner's all-purpose symbolic instruction code.

basic input/output system (BIOS). Provides the device level control of the major I/O devices in a computer system, which provides an operational interface to the system and relieves the programmer from concern over hardware device characteristics.

baud. (1) A unit of signaling speed equal to the number of discrete conditions or signal events per second. For example, one baud equals one-half dot cycle per second in Morse code, one bit per second in a train of binary signals, and one 3-bit value per second in a train of signals each of which can assume one of eight different states. (2) In asynchronous transmission, the unit of modulation rate corresponding to one unit of interval per second; that is , if the duration of the unit interval is 20 milliseconds, the modulation rate is 50 baud.

BCC. Block-check character.

beginner's all-purpose symbolic instruction. code (BASIC) A programming language with a small repertoire of commands and a simple syntax, primarily designed for numerical application.

binary. (1) Pertaining to a selection, choice, or condition that has two possible values or states. (2) Pertaining to a fixed radix numeration system having a radix of two.

binary digit. (1) In binary notation, either of the characters 0 or 1. (2) Synonymous with bit. binary notation: Any notation that uses two different characters, usually the binary digits 0 and 1.

BIOS. Basic input/output system.

bit. In binary notation, either of the characters 0 or 1.

bits per second (bps). A unit of measurement representing the number of discrete binary digits which can be transmitted by a device in one second.

block-check character (BCC). In cyclic redundancy checking, a character that is transmitted by the sender after each message block and is compared with a block-check character computed by the receiver to determine if the transmission was successful.

Boolean operation. (1) Any operation in which each of the operands and the result take one of two values. (2) An operation that follows the rules of Boolean algebra.

bootstrap. A technique or device designed to bring itself into a desired state by means of its own action; that is, a machine routine whose first few instructions are sufficient to bring the rest of itself into the computer from an input device.

bps. Bits per second.

buffer. (1) An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. Synonymous with I/O area. (2) A portion of storage for temporarily holding input or output data.

bus. One or more conductors used for transmitting signals or power.

byte. (1) A binary character operated upon as a unit and usually shorter than a computer word. (2) The representation of a character.

CAS. Column address strobe.

cathode ray tube (CRT). A vacuum tube display in which a beam of electrons can be controlled to form alphanumeric characters or symbols on a luminescent screen, for example by use of a dot matrix.

cathode ray tube display (CRT display). (1) A device that presents data in visual form by means of controlled electron beams. (2) The data display produced by the device as in (1).

CCITT. Comite Consultatif International Telegrafique et Telephonique.

central processing unit (CPU). A functional unit that consists of one or more processors and all or part of internal storage.

channel. A path along which signals can be sent; for example, data channel or I/O channel.

characters per second (cps). A standard unit of measurement for printer output.

code. (1) A set of unambiguous rules specifying the manner in which data may be represented in a discrete form. Synonymous with coding scheme. (2) A set of items, such as abbreviations, representing the members of another set. (3) Loosely, one or more computer programs, or part of a computer program. (4) To represent data or a computer program in a symbolic form that can be accepted by a data processor.

column address strobe(CAS). A signal that latches the column addresses in a memory chip.

Comite Consultatif International. Telegrafique et Teleponique (CCITT) Consultative Committee on International Telegraphy and Telephone.

computer. A functional unit that can perform substantial computation, including numerous arithmetic operations, or logic operations, without intervention by a human operator during the run.

configuration. (1) The arrangement of a computer system or network as defined by the nature, number, and the chief characteristics of its functional units. More specifically, the term configuration may refer to a hardware configuration or a software configuration. (2) The devices and programs that make up a system, subsystem, or network.

conjunction. (1) The Boolean operation whose result has the Boolean value 1 if, and only if, each operand has the Boolean value 1. (2) Synonymous with AND operation.

contiguous. (1) Touching or joining at the edge or boundary. (2) Adjacent.

CPS. Characters per second.

CPU. Central processing unit.

CRC. Cyclic redundancy check.

CRT display. Cathode ray tube display.

CTS. Clear to send. Associated with modem control.

cyclic redundancy check (CRC). (1) A redundancy check in which the check key is generated by a cyclic algorithm. (2) A system of error checking performed at both the sending and receiving station after a block-check character has been accumulated.

cylinder. (1) The set of all tracks with the same nominal distance from the axis about which the disk rotates. (2) The tracks of a disk storage device that can be accessed without repositioning the access mechanism.

daisy-chained cable. A type of cable that has two or more connectors attached in series.

data. (1) A representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by humans or automatic means. (2) Any representations, such as characters or analog quantities, to which meaning is, or might be assigned.

decibel (dB). (1) A unit that expresses the ratio of two power levels on a logarithmic scale. (2) A unit for measuring relative power. The number of decibels is ten times the logarithm (base 10) of the ratio of the measured power levels; if the measured levels are voltages (across the same or equal resistance), the number of decibels is 20 times the log of the ratio.

decoupling capacitor. A capacitor that provides a low-impedance path to ground to prevent common coupling between states of a circuit.

Deutsche Industrie Norm (DIN). (1) German Industrial Norm. (2) The committee that sets German dimension standards.

digit. (1) A graphic character that represents an integer, for example, one of the characters 0 to 9. (2) A symbol that represents one of the non-negative integers smaller than the radix. For example, in decimal notation, a digit is one of the characters from 0 to 9.

digital. (1) Pertaining to data in the form of digits. (2) Contrast with analog.

DIN. Deutsche Industrie Norm.

DIN Connector. One of the connectors specified by the DIN standardization committee.

DIP. Dual in-line package.

direct memory access (DMA). A method of transferring data between main storage and I/O devices that does not require processor intervention.

disk. Loosely, a magnetic disk unit.

diskette. A thin, flexible magnetic disk and a semi-rigid protective jacket, in which the disk is permanently enclosed. Synonymous with flexible disk.

DMA. Direct memory access.

DSR. Data set ready. Associated with modem control.

DTR. Data terminal ready. Associated with modem control.

dual in-line package (DIP). A widely used container for an integrated circuit. DIPs are pins usually in two parallel rows. These pins are spaced 1/10 inch apart and come in different configurations ranging from 14-pin to 40-pin configurations.

EBCDIC. Extended binary-coded decimal interchange code.

ECC. Error checking and correction.

edge connector. A terminal block with a number of contacts attached to the edge of a printed circuit board to facilitate plugging into a foundation circuit.

EIA. Electronic Industries Association.

EIA/CCITT. Electronic Industries Association/Consultative Committee on International Telegraphy and Telephone.

end-of-text character (ETX). A transmission control character used to terminate text.

end-of-transmission character (EOT). A transmission control character used to indicate the conclusion of a transmission, which may have included one or more texts and any assoceated message headings.

EOT. end-of-transmission character.

EPROM. Erasable programmable read-only memory

erasable programmable read-only. memory (EPROM) A storage device whose contents can be erased by ultraviolet means and new contents stored by electrical means. EPROM information is not destroyed when power is removed.

error checking and correction (ECC). The detection and correction of all single-bit, double-bit, and some multiple-bit errors.

ETX. End-of-text character.

extended binary-coded decimal interchange code. (EBCDIC) A set of 256 characters, each represented by eight bits.

flexible disk. Synonym for diskette.

firmware. Memory chips with integrated programs already incorporated on the chip.

gate. (1) A device or circuit that has no output until it is triggered into operation by one or more enable signals, or until an input signal exceeds a predetermined threshold amplitude. (2) A signal that triggers the passage of other signals through a circuit.

graphic. A symbol produced by a process such as handwriting, drawing, or printing.

hertz (Hz). A unit of frequency equal to one cycle per second.

hex. Abbreviation for hexadecimal.

hexadecimal (Hex). Pertaining to a selection, choice, or condition that has 16 possible values or states. These values or states usually contain 10 digits and 6 letters, A through F/ Hexadecimal digits are equivalent to a power of 16.

high-order position. The leftmost position in a string of characters.

Hz. Hertz.

interface. A device that alters or converts actual electrical signals between distinct devices, programs, or systems.

k. An abbreviation for the prefix kilo;that is, 1,000 decimal notation.

K. When referring to storage capacity, 2 to the tenth power; 1,024 in decimal notation.

KB (Kilobyte). 1,024 bytes.

k byte. 1,024 bytes.

kHz. A unit of frequency equal to 1,000 hertz.

kilo(k). One thousand.

latch. (1) A feedback loop in symmetrical digital circuits used to maintain a state. (2) A simple logic-circuit storage element comprising two gates as a unit.

LED. Light-emitting diode.

light-emitting diode (LED). A semi-conductor chip that gives off visible or infrared light when activated.

low-order position. The rightmost position in a string of characters.

m. (1) Milli; one thousand or thousandth part. (2) Meter.

M (Mega). 1,000,000 in decimal notation. When referring to storage capacity, 2 to the twentieth power; 1,048,576 in decimal notation.

mA. Milliampere.

machine language. (1) A language that is used directly by a machine. (2) Another term for computer instruction code.

main storage. A storage device in which the access time is effectively independent of the location of the data.

MB. Megabyte, 1,048,576 bytes.

mega (M). 10 to the sixth power, 1,000,000 in decimal notation. When referring to storage capacity, 2 to the twentieth power. 1,048,576 in decimal notation.

megabyte (MB). 1,048,576 bytes.

megahertz (MHz). A unit of measure of frequency. One megahertz equals 1,000,000 hertz.

MFM. Modified frequency modulation.

MHz. Megahertz.

microprocessor. An integrated circuit that accepts coded instructions for execution; the instructions may be entered, integrated, or stored internally.

microsecond. (μs) One-millionth of a second.

milli(m). One thousand or one thousandth.

milliampere(mA). One thousandth of an ampere.

millisecond(ms). One thousandth of a second.

mnemonic. A symbol chosen to assist the human memory; for example, an abbreviation such as "mpy" for "multiply."

mode. (1) A method of operation; for example, the binary mode, the interpretive mode, the alphanumeric mode. (2) The most frequent value in the statistical sense.

modem (Modulator-Demodulator). A device that converts serial (bit by bit) digital signals from a business machine (or data terminal equipment) to analog signals which are suitable for transmission in a telephone network. The inverse function is also performed by the modem on reception of analog signals.

modified frequency modulation (MFM). The process of varying the amplitude and frequency of the "write" signal. MFM pertains to the number of bytes of storage that can be stored on the recording media. The number of bytes is twice the number contained in the same unit area of recording media at single density.

modulo check. A calculation performed on values entered into a system. This calculation is designed to detect errors.

monitor. (1) A device that observes and verifies the operation of a data processing system and indicates any specific departure from the norm. (2) A television type display, such as the IBM Monochrome Display. (3) Software or hardware that observes, supervises, controls, or verifies the operations of a system.

ms. Millisecond; one thousandth of a second.

multiplexer. A device capable of distributing the events of an interleaved sequence to the respective activities.

NAND. A logic operator having the property that if P is a statement, Q is a statement, R is a statement, ... , then the NAND of P,Q,R,...is true if at least one statement is false, false if all statements are true.

nanosecond. (ns) One-billionth of a second.

nonconjunction. (1) The dyadic Boolean operation the result of which has the Boolean value 0 if, and only if, each operand has the Boolean value 1.

non-return-to-zero inverted (NRZI). A transmission encoding method in which the data terminal equipment changes the signal to the opposite state to send a binary 0 and leaves it in the same state to send a binary 1.

NOR. A logic operator having the property that if P is a statement, Q is a statement, R is a statement, ...,then the NOR of P,Q,R,...is true if all statements are false, false if at least one statement is true.

NOT. A logical operator having the property that if P is a statement, then the NOT of P is true if P is false, false if P is true.

NRZI. Non-return-to-zero inverted.

ns. Nanosecond; one-billionth of a second.

operating system. Software that controls the execution of programs; an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

OR. (1) A logic operator having the property that if P is a statement, Q is a statement, R is a statement, ...,then the OR of P,Q,R,...is true if at least one statement is true, false if all statements are false.

output. Pertaining to a device, process, or channel involved in an output process, or to the data or states involved in an output process.

output process. (1) The process that consists of the delivery of data from a data processing system, or from any part of it. (2) The return of information from a data processing system to an end user, including the translation of data from a machine language to a language that the end user can understand.

overcurrent. A current of higher than specified strength.

overvoltage. A voltage of higher than specified value.

parallel. (1) Pertaining to the concurrent or simultaneous operation of two or more devices, or to the concurrent performance of two or more activities. (2) Pertaining to the concurrent or simultaneous occurrence of two or more related activities in multiple devices or channels.
(3) Pertaining to the simultaneity of two or more processes. (4) Pertaining to the simultaneous processing of the individual parts of a whole, such as the bits of a character and the characters of a word, using separate facilities for the various parts. (5) Contrast with serial.

PEL. Picture element.

personal computer. A small home or business computer that has a processor and keyboard and that can be connected to a television or some other monitor. An optional printer is usually available.

picture element (PEL). (1) The smallest displayable unit on a display. (2) Synonymous with pixel, PEL.

pinout. A diagram of functioning pins on a pinboard.

pixel. Picture element.

polling. (1) Interrogation of devices for purposes such as to avoid contention, to determine operational status, or to determine readiness to send or receive data. (2) The process whereby stations are invited, one at a time, to transmit.

port. An access point for data entry or exit.

printed circuit board. A piece of material, usually fiberglass, that contains a layer of conductive material, usually metal. Miniature electronic components on the fiberglass transmit electronic signals through the board by way of the metal layers.

program. (1) A series of actions designed to achieve a certain result. (2) A series of instructions telling the computer how to handle a problem or task. (3) To design, write, and test computer programs.

programable read-only memory (PROM). Non-erasable programable memory. PROM information is not destroyed when power is removed.

programming language. (1) An artificial language established for expressing computer programs. (2) A set of characters and rules, with meanings assigned prior to their use, for writing computer programs.

PROM. Programmable read-only memory.

propagation delay. The time necessary for a signal to travel from one point on a circuit to another.

radix. (1) In a radix numeration system, the positive integer by which the weight of the digit place is multiplied to obtain the weight of the digit place with the next higher weight; for example, in the decimal numeration system, the radix of each digit place is 1.0.
(2) Another term for base.

radix numeration system. A positional representation system in which the ratio of the weight of any one digit place to the weight of the digit place with the next lower weight is a positive integer. The permissible values of the character in any digit place range from zero to one less than the radix of the digit place.

RAS. Row address strobe.

RGBI. Red-green-blue-intensity.

read-only memory (ROM). A storage device whose contents cannot be modified, except by a particular user, or when operating under particular conditions; for example, a storage device in which writing is prevented by a lockout.

read/write memory. A storage device whose contents can be modified.

red-green-blue-intensity (RGBI). The description of a direct-drive color monitor which accepts red, green, blue, and intensity signal inputs.

register. (1) A storage device, having a specified storage capacity such as a bit, a byte, or a computer word, and usually intended for a special purpose. (2) On a calculator, a storage device in which specific data is stored.

RF modulator. The device used to convert the composite video signal to the antenna level input of a home TV.

ROM. Read-only memory.

ROM/BIOS. The basic input/output system resident in ROM, which provides the device level control of the major I/O devices in the computer system.

row address strobe (RAS). A signal that latches the row addresses in a memory chip.

RS-232C. The standards set by the EIA for communications between computers and external equipment.

RTS. Request to send. Associated with modem control.

run. A single continuous performance of a computer program or routine.

scan line. The use of a cathode beam to test the cathode ray tube of a display used with a personal computer.

schematic. The description, usually in diagram form, of the logical and physical structure of an entire data base according to a conceptual model.

sector. That part of a track or band on a magnetic drum, a magnetic disk, or a disk pack that can be accessed by the magnetic heads in the course of a predetermined rotational displacement of the particular device.

serdes. Serializer/deserializer.

serial. (1) Pertaining to the sequential performance of two or more activities in a single device. In English, the modifiers serial and parallel usually refer to devices, as opposed to sequential and

consecutive, which refer to processes. (2) Pertaining to the sequential or consecutive occurrence of two or more related activities in a single device or channel.
(3) Pertaining to the sequential processing of the individual parts of a whole, such as the bits of a character or the characters of a word, using the same facilities for successive parts. (4)Contrast with parallel.

sink. A device or circuit into which current drains.

software. (1) Computer programs, procedures, rules, and possible associated documentation concerned with the operation of a data processing system. (2) Contrast with hardware.

source. The origin of a signal or electrical energy.

source circuit. (1) Generator circuit. (2) Control with sink.

SS. Start-stop transmission.

start bit. Synonym for start signal.

start-of-text character (STX). A transmission control character that precedes a test and may be used to terminate the message heading.

start signal. (1) A signal to a receiving mechanism to get ready to receive data or perform a function. (2)In a start-stop system, a signal preceding a character or block that prepares the receiving device for the reception of the code elements. Synonymous with start bit.

start-stop (SS) transmission. (1) A synchronous transmission such that a group of signals representing a character is preceded by a start signal and followed by a stop signal. (2) Asynchronous transmission in which a group of bits is preceded by a start bit that prepares the receiving mechanism for the reception and registration of a character and is followed by at least one stop bit that enables the receiving mechanism for the reception and registration of a character and is followed by at least one stop bit that enables the receiving mechanism to come to an idle condition pending the reception of the next character.

stop bit. Synonym for stop signal.

stop signal. (1) A signal to a receiving mechanism to wait for the next signal. (2)In a start-stop system, a signal following a character or block that prepares the receiving device for the reception of a subsequent character or block. Synonymous with stop bit.

strobe. (1) An instrument used to determine the exact speed of circular or cyclic movement. (2) A flashing signal displaying an exact event.

STX. Start-of-text character.

synchronous transmission. Data transmission in which the sending and receiving devices are operating continuously at the same frequency and are maintained, by means of correction, in a desired phase relationship.

text. In ASCII and data communication, a sequence of characters treated as an entity if preceded and terminated by one STX and one ETX transmission control, respectively.

track. The path or one of the set of paths, parallel to the reference edge on a data medium, associated with a single reading or writing component as the data medium moves past the component. (2) The portion of a moving data medium such as a drum, tape, or disk, that is accessible to a given reading head position.

transistor-transistor logic (TTL). A circuit in which the multiple-diode cluster of the diode-transistor logic circuit has been replaced by a multiple-emitter transistor.

TTL. Transistor-transistor logic.

TX Data. Transmit data. Associated with modem control. External connections of the RS-232C asynchronous communications adapter interface.

video. Computer data or displayed on a cathode ray tube monitor or display.

write precompensation. The varying of the timing of the head current from the outer tracks to the inner tracks of the diskette to keep a constant write signal.

Notes:

Index

Reader's Comment Form

TECHNICAL REFERENCE 6322963

Your comments assist us in improving the usefulness of our publication; they are an important part of the input used for revisions.

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions regarding the IBM Personal Computer or programs for the IBM Personal Computer, or for requests for additional publications; this only delays the response. Instead, direct your inquiries or request to your authorized IBM Personal Computer dealer.

Comments:

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

CLASS PERMIT NO. 321 BOCA RATON, FLORIDA 33432

POSTAGE WILL BE PAID BY ADDRESSEE

IBM PERSONAL COMPUTER
SALES & SERVICE
P.O. BOX 1328-C
BOCA RATON, FLORIDA 33432

Fold here

Please do not staple

Tape

International Business Machines Corporation

P.O. Box 1328-W
Boac Raton, Florida 33432

6322963

Printed in United States of America